CLASSIFICATION OF ACOUSTO-OPTIC CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING ARTIFICIAL NEURAL NETWORKS

THESIS

John W. DeBerry
Captain, USAF

AFIT/GE/ENG/89D-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/89D-10

CLASSIFICATION OF ACOUSTO-OPTIC CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING ARTIFICIAL NEURAL NETWORKS

THESIS

John W. DeBerry
Captain, USAF

AFIT/GE/ENG/89D-10

AFIT/GE/ENG/89D-10

CLASSIFICATION OF ACOUSTO-OPTIC CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING ARTIFICIAL NEURAL NETWORKS

THESIS

John W. DeBerry
Captain, USAF

AFIT/GE/ENG/89D-10

AFIT/GE/ENG/89D-10

CLASSIFICATION OF ACOUSTO-OPTIC CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING ARTIFICIAL NEURAL NETWORKS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

John W. DeBerry, B.S.,B.S.E.E.
Captain, USAF

May, 1989

Acknowledgments

There are many people to whom I owe a debt, with regard to the development and completion of this thesis. First, I acknowledge the sacrifices made by my wife, Alice, and my children, Chapele, and Sean. They shared the burden of the long hours spent doing this thesis. I am indebted to Dr. David M. Norman, Dr. Steven K. Rogers, and Dr. Mark E. Oxley for the knowledge, advice, and time they willingly shared to guide me through this research. I also wish to thank Captain Gregory L. Tarr for his help in setting up and using his NeuralGraphics simulator on the SUN 4 workstation.

I would also like to thank Dr. Matthew Kabrisky and Squadron Leader Alan P. Callaghan, Royal Australian Air Force, for their encouragement and acting as sounding boards for a great number of ideas. Finally, a special thanks to Dr. Charles Garvin, Harry Diamond Laboratories. He worked many long hours producing the correlation signature data files used in this thesis.

John W. DeBerry

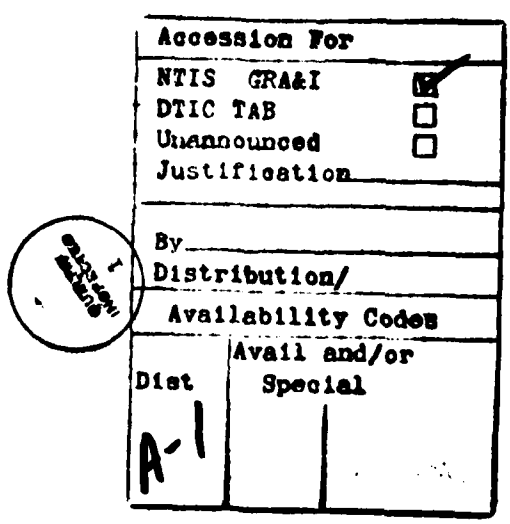

Table of Contents

List of Figures

List of Tables

Abstract

The primary goal of this research was to determine if Artificial Neural Networks (ANNs) can be trained to classify the correlation signatures of direct sequence and frequency-hopped spread-spectrum signals. Secondary goals were to determine (1) if network classification performance can be modeled with a conditional probability matrix, (2) if the symmetry of the matrices can be controlled, and (3) if using a majority vote rule over independently trained networks improves classification performance.

Correlation signatures of the spread-spectrum signals were obtained from United States Army Harry Diamond Laboratories. The signatures were preprocessed and separated into various training and testing data sets. Thirty samples of network responses for several sets of trainir ; conditions were gathered using a neural network simulator.

ANNs trained directly on correlation signature data yielded classification accuracies on test data at or near 80%. The probability matrices were stationary with regard to test sets and the ability to shift the symmetry of the matrices was demonstrated. Improvement of classification accuracy via majority vote was possible if the nets were trained on different data sets. An average improvement of 1.8% was found to be statistically significant for $\alpha = 0.05$. A metric was developed to estimate the similarity of the solutions found by networks in a given training run.

CLASSIFICATION OF ACOUSTO-OPTIC CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING ARTIFICIAL NEURAL NETWORKS

I. Introduction

1.1 Historical Background.

Over the past two decades, use of spread-spectrum systems in the Department of Defense has become increasingly common. This is primarily due to the militarily desirable antijam (AJ), antiinterference, and low probability of intercept (LPI) characteristics of these signals [1]. It is reasonable to assume that our adversaries will use spread-spectrum systems in any future conflict. Naturally, it follows that we should prepare to defeat the AJ and LPI characteristics of these signals in order to deny our enemy the free use of the electromagnetic spectrum.

In order to disrupt a hostile communications signal, we must be able to detect and classify the signal. Much of the current research is focused on solving this problem. Toward that end, researchers at the U.S. Army Harry Diamond Laboratories have developed a one-dimensional time-integrating acousto-optic (AO) correlator capable of 10^{15} operations per second [2]. The device performs the correlation transform

several orders of magnitude faster than even the most powerful digital computers. This correlator can be used in an intercept receiver to detect and capture the correlation signatures of spread-spectrum signals. Currently, the output of the correlator is examined directly by a human operator or digitized and run through curve-fitting routines on digital computers. The objective is to quickly recognize the correlation signature and obtain information about its time domain modulation characteristics.

Researchers at U.S. Army Harry Diamond Laboratories have suggested using Artificial Neural Networks (ANNs) to classify spread-spectrum communications signals. The ANN would be trained to recognize and classify specific input features present in the correlation signatures of several types of spread-spectrum signals. These features correspond to time domain characteristics of the signal. Classification of the signals in this manner essentially extracts information about the modulation parameters used to construct the signal. In an eventual hardware implementation, the network would operate in real time at the output of an AO correlator. In a detection scenario, a priori known features would be used to train the net. The net would send an alarm upon detection of those features. The package would then be mounted in a remotely piloted vehicle (RPV) and flown over an area of interest as shown in Figure 1.1. A first step in this effort would be to construct and train an ANN to determine whether a particular input spread-spectrum correlation signature is a direct sequence (DS) or frequency-hopped (FH) signal.

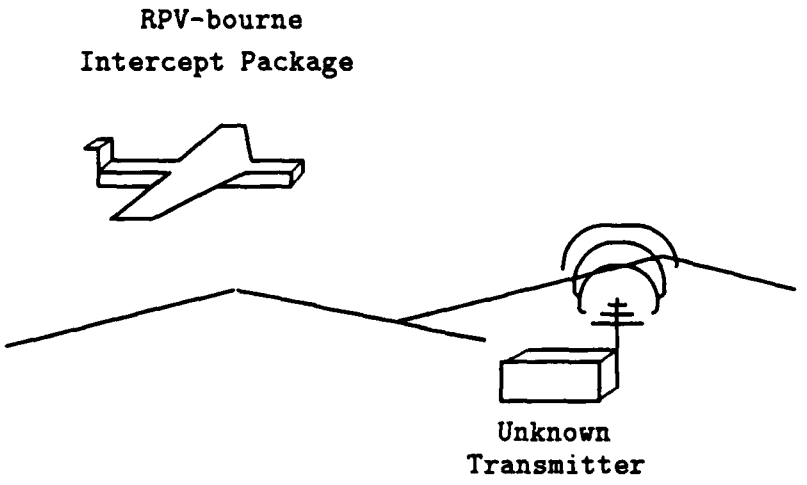

Figure 1.1. Detection Scenario [2]

1.2 Problem Statement.

The primary objective of this thesis is to answer several questions:

1. Can ANNs be trained to classify DS and FH correlation signatures, and if so, at what level of classification performance?
2. Can a trained ANN's response to previously unseen signatures be accurately modeled and described by a transition probability matrix similar to those used to describe communication channels? If so, can the symmetry of these matrices be controlled? Control of matrix symmetry would allow network classification responses to be tailored for a particular application.

3. Can classification accuracy be improved by using a majority vote decision rule over the response of an appropriate number of ANNs trained to classify the same correlation signatures?

1.3 Scope.

The intent of this thesis effort is to determine the applicability of ANN technology to the classification of spread-spectrum correlation signatures. The classification will be performed by means of ANN simulation software. The simulated ANN will be trained to classify DS and linearly stepped FH correlation signatures. The results of training will be evaluated by observing the ANN response to test data. The second question posed in the problem statement presumes a positive answer to the first question. Once ANNs are successfully trained to classify correlation signature data, an examination will be performed of the transition probability matrix model. If trained ANNs can be satisfactorily described by transition probability matrices, then an attempt to control the resulting symmetry will be made. Finally, the performance of composite networks composed of three single nets using a majority vote rule will be compared to the performance of individual nets.

1.4 General Approach.

The approach to answer the questions posed in the problem statement was broken down into three phases: basic research and experiment set-up, collecting experiment data, and analysis of the data. The first phase involved a review of

current literature and research in the area of ANN theory and applications, selection of appropriate ANN simulation software implementing the most promising topology and training algorithm, and preparation of the input data sets for use with the selected simulation software. The second phase consisted of training and performance testing a number of ANNs under various conditions with correlation signature data sets. In the final phase of the thesis effort, the performance data is analyzed and final conclusions drawn. Conclusions and recommendations will be based on observation and analysis of experimental results.

1.5 Thesis Organization.

This chapter served as an introduction and general overview of the thesis effort. Chapter two provides a discussion of the fundamental concepts of ANNs and a review of recent ANN research. Chapter three contains a complete description of resources used, definitions and notation, and how the experiments were constructed and performed. Chapter four presents the results and analysis of the experiments described. The fifth and final chapter contains specific conclusions along with recommendations for future research.

II. Background Material

2.1 Introduction.

This background material is limited to discussion of work relevant to the specific problem of applying ANNs as classifiers for continuous valued input vectors or waveforms similar to the output of the AO correlator. The material is divided into two distinct parts. The first part introduces the basic concepts and terminology commonly used in the ANN literature. The discussion provides only what is necessary to understand the second part, which presents summaries of the major findings of more recent work in the application and enhancement of ANNs. For a more detailed study, the reader is referred to the cited works.

2.2 Basic Concepts of ANNs.

An excellent beginner's guide to ANNs can be found in an article by Lippmann [3], which discusses the basic concepts of ANN models and describes six common architectures. Two of these models, the Kohonen self-organizing feature map (Kohonen net) and the multi-layer perceptron, can be applied to the problem at hand because they accept continuous valued inputs. In this thesis , the focus is on the multi-layer perceptron. According to Lippmann, ANN models reflect an extremely simplified model the current understanding of how biological nervous systems work. He states that an ANN model is fully specified by the computational characteris-

tics of its basic computing unit, network topology, and learning algorithms used for training the net [3:4, 6]. The following sections discuss these parameters as they apply to the multi-layer perceptron.

2.2.1 The Node. The fundamental computing unit of an ANN is the node, which is analogous to the neuron in biological nervous systems. The computing power of a node is fairly limited. Lippmann expresses its output as a nonlinear function of the weighted sum of its inputs, as follows:

$$Y = f(\sum_{i=0}^{N-1} X_i W_i - \theta) \tag{2.1}$$

where

Y = output
W_i = connection weights
X_i = inputs
θ = threshold

Depending on the application, the nonlinear function may be a hard limiter, threshold logic, or a sigmoid, as shown in Figure 2.1. It should be noted here that a multi-layer perceptron using a back propagation training algorithm requires a continuously differentiable function [3:17]. Typically this is a sigmoid function, also known as a squashing function, of the form:

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}} \tag{2.2}$$

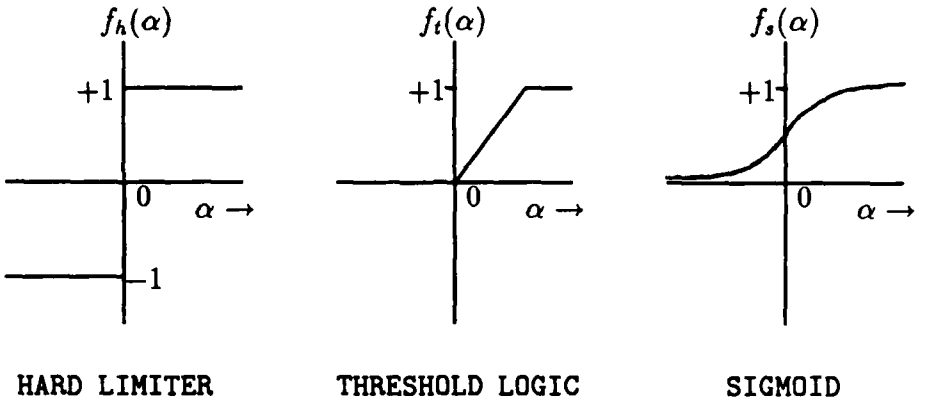

Figure 2.1. Nonlinear Functions [3:5]

where α is the argument of the function in Equation (2.1). The schematic representation of Equation (2.1) is shown in Figure 2.2.

2.2.2 Topology. When many nodes are massively interconnected in parallel or in layers, the network as a whole is capable of performing complex computations at high speeds. The way in which nodes are interconnected defines a topology. We are particularly interested in the topology of multi-layer perceptrons.

A multi-layer perceptron is composed of one or more layers of nodes between the input and the output layer of nodes. These inner layers are referred to as hidden layers. Each node in a layer is interconnected to all the nodes in the preceding and following layers. Thus, the node receives input from each of the nodes in the previous layer (or from the net inputs) and sends its output to all nodes in the next layer, as shown in Figure 2.3. Although it is possible to have more, the usual number of

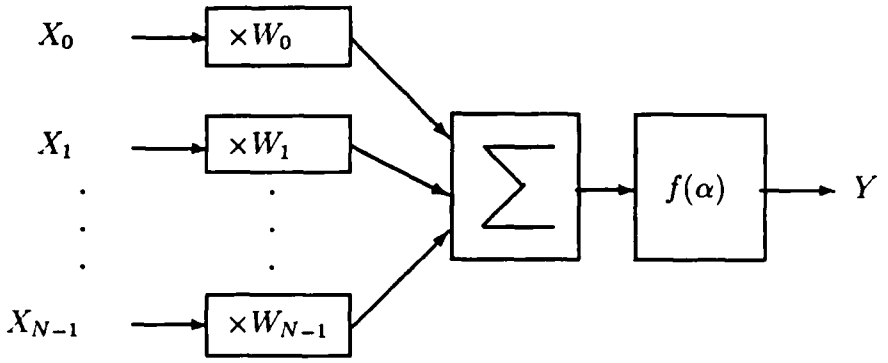

Figure 2.2. Computational Capability of a Node [3:5]

layers is three. According to Lippmann [3:16], " ... no more than three layers are required in perceptron-like feed-forward nets because a three-layer net can generate arbitrarily complex decision regions." It can be shown that two layer networks also have this ability, but may require more training iterations to reach an equivalent solution [4].

2.2.3 Learning Algorithms. When individual nodes are interconnected and organized into a network topology, the network can be trained with a learning algorithm to perform a specific task. The learning algorithm specifies the way in which the weights between nodes are updated during training. Initially, the weights of an untrained net are set to small random values. A number of input examples, referred to as a training data set or training set exemplars, are presented to the network one at a time. At each presentation, an error signal is generated and the weights updated so that the error is minimized via a gradient search. Training continues until the

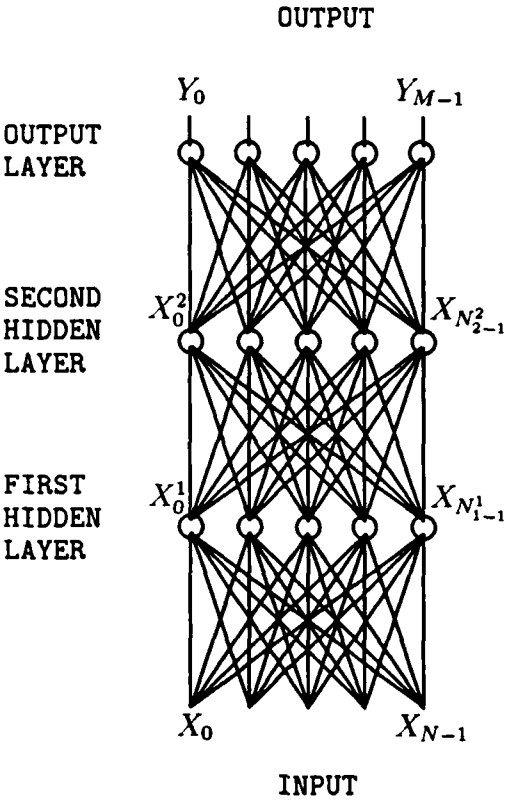

Figure 2.3. Three-layer Perceptron [3:16]

weights no longer change or the change is less than some threshold value. At this point training is terminated [3]. The heart of the training algorithm is the way the error signal is computed and minimized.

Training algorithms can be separated into three categories: unsupervised, supervised, and self supervised. In unsupervised training, the network is given no information as to what class the input belongs to. The error signal is computed solely as a function of the current input and output. On the other hand, in supervised learning, the net is provided information about the correct classification (the

desired output) for the present input. Thus, the error signal is a function of the current network output and the desired output [3:19]. In self-supervised learning, the network monitors its performance internally, feeding back an error signal to itself [5].

The multi-layer perceptron is trained with supervision using the back propagation gradient algorithm, which is a generalized form of the least mean square (LMS) algorithm. Because of the multiple layers, an error signal for each layer must be produced. For the output layer the error signal is given by:

$$\delta_j = Y_j(1 - Y_j)(d_j - Y_j) \tag{2.3}$$

where Y_j is the output of node j, and d_j is the desired output of node j. Equation 2.3 is the partial derivative of the error with respect to the output layer weights [4]. For the hidden layers, the error is computed a little differently, since there is no way to specify the desired output of hidden nodes. An error signal for a hidden layer is given by:

$$\delta_j = X'_j(1 - X'_j) \sum_k \delta_k W_{jk} \tag{2.4}$$

where X'_j is the output of node j (or input j), and k ranges over all nodes in the layer above. The weights and error signals are computed and updated from the output layer back to the input in a recursive fashion. The weights in each layer are updated

according to:

$$W_{ij}(t+1) = W_{ij}(t) + \eta\delta_j X'_i + \alpha(W_{ij}(t) - W_{ij}(t-1)) \tag{2.5}$$

where,

W_{ij} = weight from hidden node (or input) i to node j
X'_j = output of node j (or input j)
η = learning rate
δ_j = error term for node j
α = momentum gain

The learning rate, η, controls how fast the weights converge. The momentum gain, α, weights the contribution of the previous update to the current update. Both of these parameters are set to a value between 0 and 1. Multi-layer perceptrons trained with the back propagation algorithm can be used to determine which class an unknown input is most similar to. The input to the trained network may be corrupted by noise or in some way different than the inputs used for training. In either case, the network must classify an input which is not exactly the same as the inputs used to train the network [3].

2.3 Current Research in ANNs.

The following paragraphs summarize some of the recent research efforts in which ANNs were used to classify real world, continuous valued data. The discussions are limited to presenting the purpose or objectives of the research, a general description of the experiment, and the overall results or major findings. The

research efforts are divided into those which were application oriented and those that were enhancement oriented. The application efforts included experiments in training three-layer perceptrons with the back propagation algorithm as described in Section 2.2.3. The enhancement efforts were directed more towards improving the training performance by modifying the back propagation algorithm or network structures.

2.3.1 Application Oriented Research. A great deal of ANN research in the military community is directed towards the problem of recognition and classification of targets from sensor data. Successful development of this capability would be the first step in constructing autonomous weapons systems. Troxel [6] and Gorman [7] conducted research in this area.

Troxel trained a three-layer perceptron to classify multi-function laser radar data of tanks and trucks at various aspect angles. His approach was to first obtain segmented target images using a doppler segmenter developed by Ruck [8]. The segmented images were then transformed into a position, scale, and rotation invariant (PSRI) feature space. A correlation was performed between the transformed data and the feature space itself. The correlation peak was found and a window of 49 data points around the peak was extracted for classification. Once these data points were normalized, Troxel [6:I-594] said the data could "...be thought of as a 49 dimensional vector of length 1." These vectors were then used to train the networks. Troxel reported a maximum classification accuracy of 80% on test data sets. In

addition, he suggested a procedure for selecting an appropriate number of nodes for each hidden layer and observed that greater numbers of training vectors would be needed to ensure good classification performance of real world data.

Similar experimental work done by Gorman [7] involved the classification of sonar returns from undersea objects. The primary objective of Gorman's experiments was to determine if an ANN could learn to distinguish the sonar returns from a metal cylinder (target) from the returns from a cylindrical rock (non-target). The raw data for the experiment were spectrograms of the sonar returns of the cylinder and rock at various aspect angles, where aspect angle refers to the angle from which the objects were illuminated with the sonar pulse. The sonar returns were transformed into a spectral envelope representation by integrating over sampling apertures of the short-term Fourier transform spectrogram. Sixty samples of the spectral envelope were normalized and used as input to the network. Two layer perceptrons with various numbers of hidden nodes were trained and performance tested. The networks were trained on two types of data sets: aspect-angle independent and aspect-angle dependent. In the former case, the sonar returns used for training were selected at random without regard to aspect angle. In the latter case, the returns used for training were chosen so as to ensure that all aspect angles were represented. The best test set classification accuracy, 90.4%, was yielded by a network with 12 hidden nodes and trained on the aspect-angle dependent training set. In general, Gorman found that greater numbers of hidden nodes reduced performance variations, and training

with aspect-angle dependent data sets resulted in the best classification accuracy. In other words, for best results, make sure the training set contains examples of all the various *flavors and colors* of the things to be recognized.

2.3.2 Enhancement Oriented Research. Several of the most recent ANN research efforts conducted at the Air Force Institute of Technology (AFIT) involved exploration of methods to improve network training performance. Training performance is defined by the number of training iterations required for the network weights (and thus its classification performance) to converge to relatively constant values. The classification accuracy of a network is simply the percent of correct responses to a set of inputs. Usually, a test set of exemplars, not included in the training set, is used for this purpose. The various approaches discussed in the following paragraphs include:

1. modifying error minimization algorithms.
2. modifying the error signal.
3. modifying the computational functions of the network nodes.
4. combining two network structures and algorithms into one larger network.

Each of these approaches was recently investigated in thesis efforts at AFIT. In addition, the performances of the enhanced networks were compared to the performances of the basic three-layer perceptron trained with the first order back propagation gradient method described in Section 2.2.3. All of the networks were eventually trained

on the Ruck [8] data sets, thus establishing a common reference for judging the algorithm modifications.

Piazza [9] performed experiments training three-layer perceptrons using a second order error minimization technique in the back propagation algorithm. He also trained networks with the first order methods: the basic back propagation gradient method, and back propagation with a momentum term (momentum method). In the gradient method, α in Equation 2.5 is set to zero. Piazza found training performance of the momentum and second order methods significantly better than that of the gradient method. The second order method was only slightly better than the momentum method. Average test set classification performances were 75%, 78%, and 78% for the gradient, momentum, and second order methods, respectively.

Several approaches to improving network training performance were demonstrated by Lutey [10]. He attacked the problem in three different ways: modifying the error generating function, varying the rise rate of the sigmoid, and implementing more complex weighting functions in the nodes. Networks were trained with each of the proposed improvements and their performances compared to the baseline performance of the gradient method. As before, all networks were trained on the Ruck data. All three techniques showed significant improvements in training performance by reducing the iterations required for convergence over the baseline case.

One final technique for enhancement was suggested by Tarr [11]. He performed an experiment in which two different network structures, a Kohonen net and multi-layer perceptron, were combined into one network. The Kohonen net served to organize and simplify the input data. The outputs of the Kohonen net were then fed to the perceptron which performed the classification. Again, the result was a significant reduction in training iterations to reach convergence. The hybrid network had many more nodes than a three-layer perceptron back propagation net of equivalent classification performance. Tarr observed that it appeared that the hybrid net was trading a reduction in training time (or iterations) for numbers of nodes. The test set classification performance accuracy for the best case hybrid network was 74%, which was essentially equivalent to the baseline gradient back propagation perceptron. In addition, Tarr observed that hybrid networks performed better than multi-layer perceptrons when there were ambiguous decision regions in the data set, and vice versa for unambiguous data sets.

2.4 Summary.

The problem at hand is to determine if ANNs can be used to classify the correlation signatures of spread-spectrum communications signals. The literature examined in previous sections indicates that it should be possible to train a three-layer perceptron to yield a classification accuracy in the range of 75% to 90% using any one of the techniques or algorithms described. Also, in most cases, some sort of data transformation and/or normalization was performed on real world data before

presentation to an ANN. Finally, it is apparent that most researchers average the network performances over a number of training trials. This immediately implies that random processes are at work and that statistical analysis of the distributions of performance would be the appropriate analysis tool.

III. Methodology

3.1 Introduction.

This chapter provides information concerning the details of how the experiments were performed. First, a description of the resources used is presented. This is followed by a description of how each of the experiments was designed and implemented. Finally, the manner in which the results of the exeriments were analyzed is given.

3.2 Resources.

This section will cover the descriptions of all the resources used to accomplish the experiments performed in this thesis. The following paragraphs describe the spread-spectrum correlation product data files, the artificial neural network (ANN) simulator, and the preprocessing performed on the correlation product data in order to use it with the simulator.

3.2.1 The Correlation Product Data. The spread spectrum correlation signatures were obtained from the sponsor of this thesis, Harry Diamond Laboratories (HDL). The signatures were generated by simulating various direct sequence (DS) and frequency hopped (FH) signals and feeding them into an acousto-optic (AO) correlator. Different signatures of each type of signal were generated by varying several modulation parameters; chip rate, carrier frequency, psuedo-random code, etc.

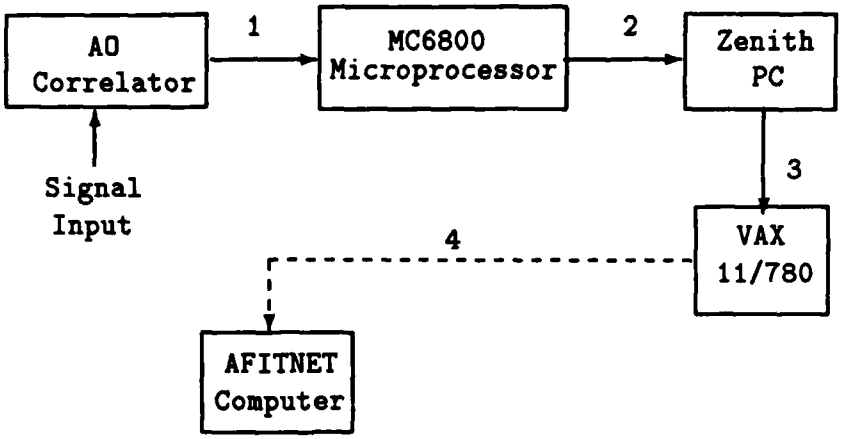

Figure 3.1. Path of Correlation Product Data. (1) AO correlator output (2) output of MC6800 microprocessor (3) upload files to mainframe (4) files transferred across MILNET to AFIT [2]

The FH signals were not driven by a psuedo-random code, but were stepped across frequency ranges by a linear stepper. For the remainder of this thesis, exemplars derived from DS signatures will be known as class 1 exemplars, while those derived from FH signatures will be class 2 exemplars.

The output of the AO correlator for a given signature was sampled and written to an ASCII file as a column of numbers. These files all contained 1,000 data points. The files were transmitted from HDL to an AFIT computer via MILNET using the DoD file transfer protocol (FTP). Figure 3.1 shows a block diagram of the process. The files were then downloaded to a personal computer for the preprocessing into the format required by the ANN simulator.

3.2.2 The ANN Simulator. The ANN simulator chosen for this thesis was the NeuralGraphics simulator written by Captain Greg Tarr [11]. The simulator was originally designed to run on a Silicon Graphics IRIS workstation. After a minor modification to turn off the graphics display, the software was ported to and run on a SUN 4 workstation. Two other minor modifications were made to support the data requirements of the experiments. First, a terminal test routine was added. The routine writes the network response to the test data set to a file. The file contains a test exemplar label, the true class, and the network's classification decision for each exemplar in the test data set. In addition, a training history file for the networks was also written to a file. The history files contain the outcome of performance tests after every 1,000 training iterations until training was terminated. A sample of both types of these files can be found in Appendix B. The second modification to the software concerned the setting of the seeds for the random number generator. The generator is used to select values for the initial weight values of the nets at the start of training and to randomly select an exemplar for presentation to the net during training. These training conditions were controlled by varying how and when the generator seed was selected. For example, when it was desired that successive nets be trained from the same initial weight state, the seed for selecting the initial weights was set to the same arbitrary constant for each net trained. If it was desired that successive nets be trained from different initial weight states, the seed was set to the

current value of the system real time clock. Control over the order of presentation of training exemplars was obtained in a similar manner.

The NeuralGraphics simulator allows the user to select one of several network structures and training algorithms. The back propagation networks all use the update rules described in Section 2.2.3. The values of the learning rate η was 0.3, while the momentum gain α was 0.8. The simulator also allows user specification of the number of nodes in the hidden layers of the network. As pointed out by Piazza [9], Tarr [11], and Troxel [6], there is no known method for selecting the best number and arrangement of hidden layer nodes. In short, one usually resorts to a trial and error search for a combination that yields reasonably good results. Preliminary training runs showed that the networks trained relatively well over a wide range of combinations and numbers of hidden layer nodes. Of the various combinations tried, ten nodes in the second hidden layer and 18 nodes in the first hidden layer appeared to yield the most consistent classification performance. In addition, it was observed that, usually, no improvement in performance occurred after 20,000 training iterations. These above mentioned parameter values were used for training all networks in this thesis and 20,000 iterations was used as the condition to terminate training. For further information on the NeuralGraphics simulator, the reader is referred to [11].

3.2.3 Data Set Construction. As previously stated, the correlation signature data files contain 1,000 data points and are not normalized. The number of out-

put nodes and input nodes of a network is determined by number of classes to be recognized and the number of elements in the input exemplar. Since presentation of all 1,000 elements of a data file as inputs to a network was beyond the capacity of the simulation software, some preprocessing was necessary. All of the data files containing the correlation signatures were processed in the following manner. First, the data files were reduced to 500 points by averaging consecutive pairs of data points together. Next, the data point with the largest absolute value was found and all data points divided by this value. The result of these two steps was a 500 point signature pattern linearly compressed to values between -1 and 1. The final step was to extract a 50 point window centered about the peak positive value of the correlation product pattern. The position of the maximum positive value was found and the 50 data points roughly centered on this position were extracted and written to a file. A total of 101 class 1 and 108 class 2 exemplar patterns were available to train the networks. The data sets used by the simulator to train networks for the different experiments were constructed from this pool of exemplars. The details of the construction of the data sets may be found in Appendix B.

3.2.4 Definitions and Notation. Before proceeding with the descriptions of the experiments, the terms and notation used in the remainder of this thesis must be defined. There are several metrics used by the NeuralGraphics simulator to evaluate network performance: total error, *right* classification, and good classification. The

error for a given exemplar is defined as

$$error = (\sum_{i=1}^{m}(d_i - y_i)^2)^{1/2} \tag{3.1}$$

where m is the number of output nodes, d is the desired output, and y the actual output. The total error over the entire training set is just the sum of the error for each training exemplar. The network yields a *right* classification if the error at each output node, for that exemplar, is less than 0.2. A *good* classification only requires that the maximum output occur at the node representing the class of the input exemplar [11]. For example, if the desired output for a given exemplar were [1, 0, 0], then [0.91, 0.09, 0.09] would be a *right* classification and [0.7, 0.5, 0.4] would be a *good* classification. The percentage of *right* and *good* classifications are calculated for both the training set exemplars and the test set exemplars. As previously mentioned, these five performance metrics are computed after every 1,000 training iterations.

Since the output y is a function of the inputs and the set of weights, w_{ij}, then for a fixed set of training inputs, it is quite natural to envision an *error surface* generated by allowing each weight to vary over its entire range of possible values. The result is an n dimensional error surface for the given set of training exemplars, where n is the total number of weights in the network. It is the global minimum on this surface that the back propagation algorithm seeks to find: in other words, the specific combination of the n weights that yields the lowest total error. For

further information on this concept, Tarr [11] provides a very good discussion and demonstration of the two dimensional case.

As will be seen in the next section, the training history of each performance metric mentioned will be examined for sets of networks trained under various conditions. However, the bulk of the experiments examine the effect of the training conditions on only one performance measure after completion of training. The performance metric used is the percent of *good* classifications on test data set exemplars. Furthermore, in this thesis, this metric will be viewed in probabilistic terms. For example, assume a certain trained network is tested with a data set having 50 class 1 exemplars and 50 class 2 exemplars. If the network yields a *good* classification for 80 of the 100 test exemplars, this will be expressed as $P(\text{good}) = 0.80$. Also, if the network yields *good* classification on 35 of the 50 class 1 exemplars and 45 of the 50 class 2 exemplars in the test data set, then these observations will be expressed as the conditional probabilities, $P(1 \mid 1) = 0.70$ and $P(2 \mid 2) = 0.90$. It should be clear that conditional probabilities for incorrect classification are $P(2 \mid 1) = 1 - P(1 \mid 1)$ and $P(1 \mid 2) = 1 - P(2 \mid 2)$. For this example, the proportions of the class 1 and class 2 exemplars in the test set can be expressed as the a priori probabilities $P(1) = P(2) = 0.50$. The characterization of network performance in this manner is similar to the way information channels are characterized in the communications field. The conditional probabilities are known as channel transition probabilities and collectively referred to as the transition probability matrix or $\underline{P}$ matrix. Networks

trained for a two class recognition problem are modeled as a binary channel. For further information regarding channel models, the text by Hamming [12] is an excellent choice. Characterization of network performance with this model was inspired by the *confusion matrix* found in the work of Piazza [9]. These confusion matrices were simply counts of how the input exemplars of each class were distributed over the output classifications by the network. If the number of test exemplars is sufficiently large, it is not hard to extend the confusion matrix concept to the $\underline{P}$ matrix described above.

Finally, a few words regarding the naming conventions used in the rest of this thesis are in order. Since there are random processes involved in training networks, the value of the P(good) is a random variable with some probability density function (PDF), and will have some distribution about a mean. Examination of the effects of different training conditions or test conditions must be done by examining the the differences in the distributions of P(good). A sample of these distributions, or PDFs, will be obtained by training a number of networks for a given set of conditions and observing the outcomes. In the next section, a total of five sets of training conditions will be specified. In addition to this, there will be three different methods for generating distributions of interest. Specifically, the $P(1 \mid 1)$, $P(2 \mid 2)$, and P(good) distributions for (1) observed outcomes of individual networks, (2) outcomes of majority vote networks constructed from observed outcomes of three individual networks as shown in Figure 3.2, and (3) the calculated outcomes of

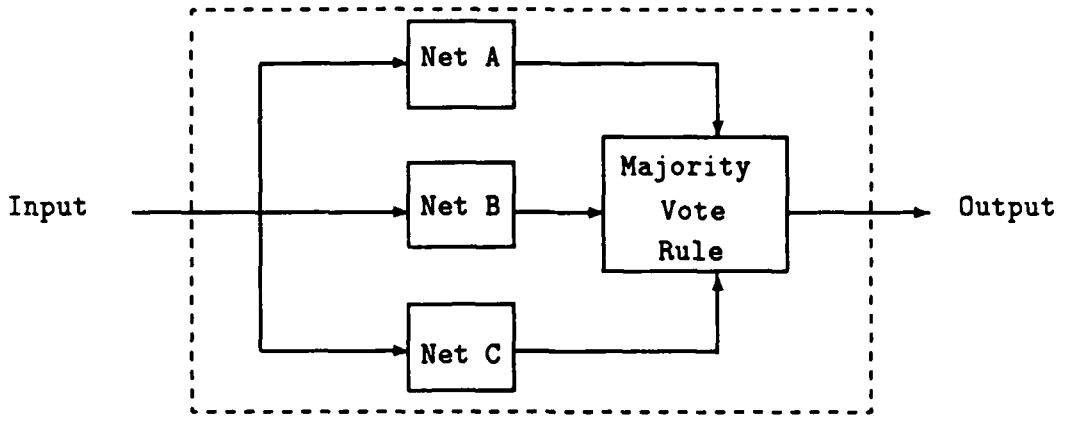

Figure 3.2. Diagram of Majority Vote Networks

majority vote nets constructed using the probability matrices of three individual nets. As one can see, quite a number of different PDFs will need to be named and referred to. The following shorthand naming conventions shall be used. The letter R followed by a single digit will specify a set of training conditions for a training *run*. The letter S will indicate the distribution is for *single* nets. The letter M will signify the distribution is for *majority vote* nets. The appearance of P11, P22, or PG will designate the distributions as $P(1 \mid 1)$, $P(2 \mid 2)$, or $P(\text{good})$, respectively. A letter C prepended to the name indicates a *calculated* distribution. As an example, the string R1SP11 refers to the $P(1 \mid 1)$ distribution for the single nets of Run 1. Similarly, the string CR5MPG refers to the calculated $P(\text{good})$ distribution for majority vote networks of Run 5.

3.3 Experiment Design.

All of the experiments defined in this section are designed to examine the influence that various conditions have on the *good* classification performance metric. For each experiment, the purpose or intent, network training and testing conditions, and expected results will be specified. Additionally, the data requirements and distribution nomenclatures will also be given.

3.3.1 Training Performance. While not an actual experiment in the true sense of the word, the documentation of how the networks train to their final states is of general interest. It can guide future research using the same data sets by documenting a baseline performance for comparisons. The raw data required for documenting the training history of the performance metrics were the history files of nets trained in the runs to be specified in the following sections. Each performance metric will be averaged over 30 nets trained in a given run, at 1,000 iteration intervals. These averages will be plotted against iterations to yield a training performance curve.

3.3.2 Characterization of ANN Performance with the $\underline{P}$ Matrix Model. In this experiment, the validity of using a $\underline{P}$ matrix to characterize the performance of trained networks will be tested. Specifically, the stationarity of the matrix over different test data sets for networks trained in the same manner, will be examined. Thirty networks were trained using the same 102 training exemplars. The initial

weights at the start of training and the presentation order of training exemplars were different for each net trained. These networks were tested with a baseline data set having 50 class 1 exemplars and 50 class 2 exemplars. The same networks were also tested with data sets having a 40/60% mix of class 1 to class 2 exemplars. Thirty of these data sets were constructed from the baseline test set by randomly removing 12 class 1 exemplars and adding seven class 2 exemplars. The same seven class 2 exemplars were added to each test set. The results of testing the 30 nets with the baseline test set will be called Run 1, while the results of testing the 30 nets with data sets having the 40/60% exemplar mix will be called Run 1a. Note that the S designation for single nets, has been omitted in this case since there will be no majority vote nets constructed for these runs.

A total of seven distributions were needed to perform the experiment. The distributions for the conditional probabilities, $P(1 \mid 1)$ and $P(2 \mid 2)$, and the joint probability, $P(good)$, were generated for each run. In addition, a calculated $P(good)$ distribution was generated by using the $\underline{P}$ matrices of Run 1 and assuming the a priori probabilities $P(1) = 0.40$ and $P(2) = 0.60$. The nomenclatures for these distributions are; R1P11, R1P22, R1PG, CR1PG, R1AP11, R1AP22, and R1APG.

If the $\underline{P}$ matrices are stationary with respect to test data sets, then the following results should be expected. There should be no difference between the $P(1 \mid 1)$ and $P(2 \mid 2)$ distributions of either run, nor should there be any difference between the calculated $P(good)$ distribution of Run 1 and the $P(good)$ of Run 1a. The difference

between the P(good) PDFs of Run 1 and Run 1a should be due only to the change in the a priori probabilities of the exemplar classes in the test data sets.

3.3.3 Controlling $\underline{P}$ Matrix Symmetry. From the preliminary training runs, it was known that the conditional matrices for nets trained with a 50/50% mix of exemplar classes were asymmetric. The *good* classification of class 1 exemplars was much poorer than for class 2 exemplars. If this $\underline{P}$ matrix were for a communications channel, adjustments would be made so that the channel was symmetric [12:143]. Assuming it would be desirable to have a neural network with a symmetrical $\underline{P}$ matrix, a question arises: Is it possible to cause the $\underline{P}$ matrix to move toward symmetry? This experiment is designed to answer this question.

The $\underline{P}$ matrix PDFs of Run 1 will be used as the baseline case of trained network responses. Another run of nets, which shall be referred to as Run 2S, were trained with a 60/40% mix of class 1 to class 2 exemplars. The training sets were constructed by randomly removing 17 class 2 exemplars from the baseline training set of Run 1. A total of 30 of these training sets were constructed in this manner, each one slightly different with regard to the exact set of class two exemplars included. The test set exemplars for all of these nets were identical to the test set used for Run 1. The three new PDFs of interest are the $P(1 \mid 1)$, $P(2 \mid 2)$, and P(good) of Run 2S which will have the nomenclatures R2SP11, R2SP22, and R2SPG.

If the networks are trained *harder* on class 1 exemplars, as in Run 2S, then the weight space solutions found by the nets should shift to be more favorable to class

1 recognition. In terms of the PDFs, the mean value of the $P(1 \mid 1)$ PDF in Run 2 should be greater than that of Run 1, and the mean of the $P(2 \mid 2)$ PDF of Run 2 should be less than that of Run 1. It is hoped that the means of the two $P(\text{good})$ distributions are essentially the same.

3.3.4 Improvement of Classification Performance via Majority Vote Rule. This experiment is designed to determine if classification performance can be improved by using a majority vote decision rule over three separately trained nets. In addition, the conditions of training necessary to achieve this improvement will be explored. The idea proposed here is very similar to the use of redundancy in communications systems to reduce probability of symbol or bit error. Basically, for a channel having a specified probability of bit error, two redundant bits are sent for every information bit. If the cause of corruption (noise) is independent and uncorrelated during each successive bit transmission, then the joint probability that two or all three bits are in error will be much less than that of any single bit. However, in order to realize the improvement, there must be independence from one trial to the next. It is obvious that the solutions found by the successively trained networks are not totally independent; the back propagation algorithm is seeking the same global minimum on the same error surface on each trial. However, one cannot say that the solutions are totally dependent either, since it has been observed that the nets do not end up in exactly the same place in weight space [9:page 4–9].

Four runs of nets trained under different conditions were required for this experiment. The test data sets for these four runs were identical. Recall that Run 2S nets were trained such that initial weights, presentation order of training exemplars, and exact composition of the training data set were varied from one net to the next. An additional 90 nets were trained in this manner. These nets were then used three at a time to construct 30 majority vote nets as shown in Figure 3.2. The $\underline{P}$ matrices for these nets were generated by observing the decisions of each of the single nets for the test set exemplars and determining the final classification by majority vote. At the same time, a calculated majority vote matrix was generated by assuming independence and using the $\underline{P}$ matrices of the the 90 single nets three at a time. The actual comparisons will only use the P(good) PDF of each of the matrices. This same basic procedure was repeated for runs 3, 4, and 5: 120 nets were trained, 30 nets were used to construct the single net PDFs, the other 90 to construct the majority vote and calculated majority vote PDFs. The nets of Run 3 were all trained on the same data set, but the initial weights and exemplar presentation orders were different for every net. In Runs 4 and 5, the training data sets were the same as Run 3, but in Run 4, only the initial weights varied, while in Run 5, only the presentation order varied. The nomenclatures for the PDFs to be compared are: R2SPG, R2MPG, CR2MPG, R3SPG, R3MPG, CR3MPG, R4SPG, R4MPG, CR4MPG, R5SPG, R5MPG, and CR5MPG.

If there is a sufficient degree of independence in the outcomes of the single nets of any of the runs, the result should be that the average of the majority vote P(good) PDF is greater than that of the single nets. This gain in average performance should be some portion of the gain for the calculated majority vote nets. Also, comparisons between the majority vote gains of the different runs should provide some insight into the relative degree each controlled condition contributes to the randomness of the weight space solutions found by the nets.

3.3.5 Influence of Training Data Sets, Initial Weights, and Exemplar Presentation Order on Network Solutions. This experiment uses the data obtained while training the nets of runs 2 through 5. The intent of this experiment is to discover the degree of similarity of decision regions formed by nets in a given run.

For each network trained in the previous experiment, a list of the exact exemplars incorrectly classified by that net was generated. The lists for the first 30 nets trained in each run will be compiled into a master list for each run. The list will contain the file name of all the exemplars incorrectly classified by *any* of the 30 nets in a given run. In addition, the exact number of nets in the run that incorrectly classified each exemplar on the list will be recorded.

The expected results in this case can only be stated in general terms. If the decision regions formed by the nets of a given run are all relatively similar, then it should be found that a majority of the test exemplars on the list were incorrectly classified by most, if not all, of the networks trained for that run. In other words, if

two nets find exactly the same solution, they should make exactly the same mistakes. On the other hand, if two nets find equally good, but very different decision regions, then even though each net makes the same number of mistakes, the mistakes made by one may be quite different than those made by the other.

3.3.6 Summary. The previous subsections specified the purpose, training and testing parameters, and expected results of each experiment. The training runs were set up to train the networks and obtain the raw data. The raw data was then used to generate the different distributions required for the various experiments. Each training run had a unique set of training and testing parameters. Table 3.1 summarizes these parameters for all of the training runs described in the preceeding paragraphs.

3.4 Analytical Methods.

The quantities being dealt with in this thesis are samples of populations. In some cases, the samples may in fact be from the same population and in other cases, from entirely different populations. By analysis and comparison, we hope to discover which of these statements apply to the sample distributions being compared. The experiments outlined in the previous section were designed with this goal in mind. The determination of whether the PDFs being compared come from the same or from different populations will be the results from which the conclusions will be drawn. The method chosen to do this is hypothesis testing. The PDFs were tested

Table 3.1. Summary of Experiment Run Parameters

Parameter	Run Designation					
	R1	R1A	R2	R3	R4	R5
# of Nets Trained	30	*	120	120	120	120
# of Training Exemplars	102	*	85	102	102	102
Training Set Mix class 1/class 2	50/50 %	*	60/40 %	50/50 %	50/50 %	50/50 %
Identical Training Sets	Yes	*	No	Yes	Yes	Yes
# of Test Exemplars	100	95	100	100	100	100
Test Set Mix class 1/class 2	50/50 %	40/60 %	50/50 %	50/50 %	50/50 %	50/50 %
Identical Test Sets	Yes	No	Yes	Yes	Yes	Yes
Generator Seed for Initial Weights	Variable	Variable	Variable	Variable	Variable	Fixed
Generator Seed for Presentation Order	Variable	Variable	Variable	Variable	Fixed	Variable
Majority Vote Networks	No	No	Yes	Yes	Yes	Yes
Distribution Nomenclatures	R1P11 R1P22 R1PG CR1PG	R1AP11 R1AP22 R1APG	R2SP11 R2SP22 R2SPG R2MP11 R2MP22 R2MPG CR2MP11 CR2MP22 CR2MPG	R3SP11 R3SP22 R3SPG R3MP11 R3MP22 R3MPG CR3MP11 CR3MP22 CR3MPG	R4SP11 R4SP22 R4SPG R4MP11 R4MP22 R4MPG CR4MP11 CR4MP22 CR4MPG	R5SP11 R5SP22 R5SPG R5MP11 R5MP22 R5MPG CR5MP11 CR5MP22 CR5MPG

* This run tested nets produced in R1 with different test sets

using a commercial software package, *Statistix*^{tm}, from NH Analytical Software [13], running on an IBM compatible personal computer. For all comparisons equality is the standard null hypothesis. Alternative hypotheses may be inequality, less than, or greater than, as it fits the particular comparison. Additionally, the standard level of significance for all tests will be $\alpha = 0.05$. The software package provides the p–value for all tests. A p–value is the probability that the observed difference in the samples could have occurred by random chance. Thus, a very low p–value indicates that the samples are not from the same population. If the p–value falls below the α value, the null hypothesis is rejected, and one may accept the applicable alternative hypothesis.

There are many test statistics available for hypothesis testing. The parametric tests are, in general, more powerful, but require that the samples come from a normal population. Non-parametric tests only require that the observations be independent. As a standard procedure, the results will include a test for normality using the Wilk-Shapiro test statistic [14]. A table of the percentage points for this test statistic has been provided in Appendix C. Whenever the assumption of normality holds, a parametric Two-Sample t-test will be performed, otherwise, non-parametric tests will be used. The non-parametric tests available are; the Wilcoxon Signed Rank Test, Rank Sum Test, Kruskal-Wallis One Way Analysis of Variance (AOV), and the Median Test. For further information on hypothesis testing or test statistics, refer to [13] [15] [16].

IV. Results

4.1 Introduction.

This chapter documents the major results of this thesis effort. First the average training performance histories are presented. These performance curves show how the networks reached their final states. The final states are more closely analyzed in the remaining sections covering comparisons of the various output probability density functions (PDF) of populations obtained from net responses to test set data.

4.2 Training Performance.

There are several metrics used to evaluate the training performance of neural networks: the error over the training data, percent of *right* and *good* classifications over the training data, percent of *right* and *good* classifications over the test data. These metrics are usually used to judge the worth of different training algorithms or data sets. All of these metrics were obtained for 30 nets trained in a given run. The training histories for Run 1 were not recorded because these nets were trained under the same conditions as the nets of Run 3. The histories for runs 4 and 5 are not shown due to the fact that they parallel the history for Run 3 so closely, that it becomes difficult to distinguish one from the other on the plots. The following sections discuss the training history of each metric for Run 2 and Run 3.

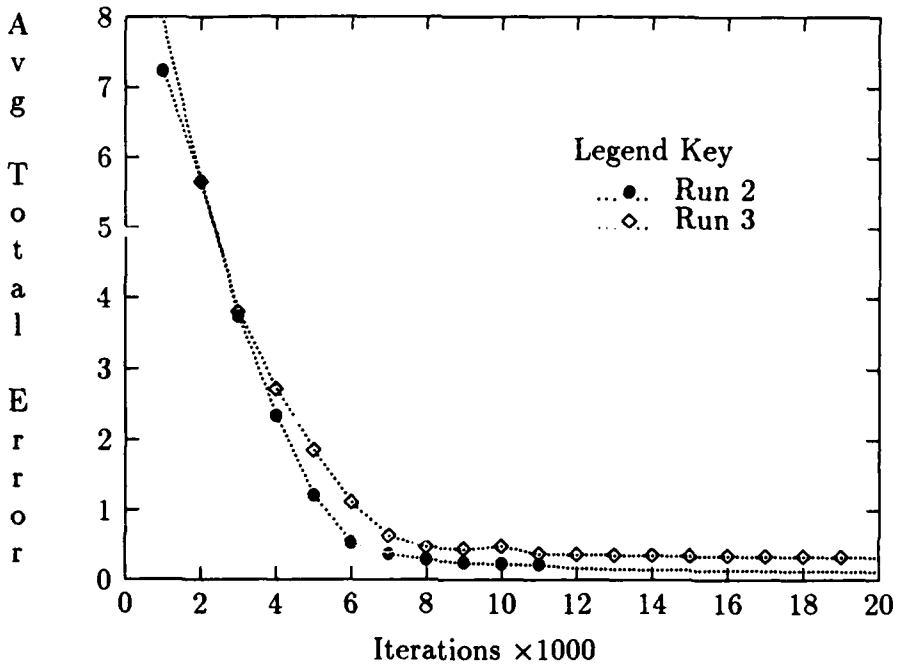

Figure 4.1. Training Histories for Average Total Error

4.2.1 Average Error History. The average error histories over 30 nets trained in Run 2 and Run 3 are shown in Figure 4.1. The figures show that by 12,000 iterations the error was well past the knee on both curves and appears to be asymptotically approaching zero. Note that the curve for Run 2 converges to zero slightly faster than Run 3. No analysis was performed to determine if this difference was significant.

4.2.2 Average Right *Classification Histories on Training Data.* The histories of average percent *right* classification on training data for Run 2 and 3 are shown in Figure 4.2 . After 1,000 iterations, the nets of both runs average close to 10% and

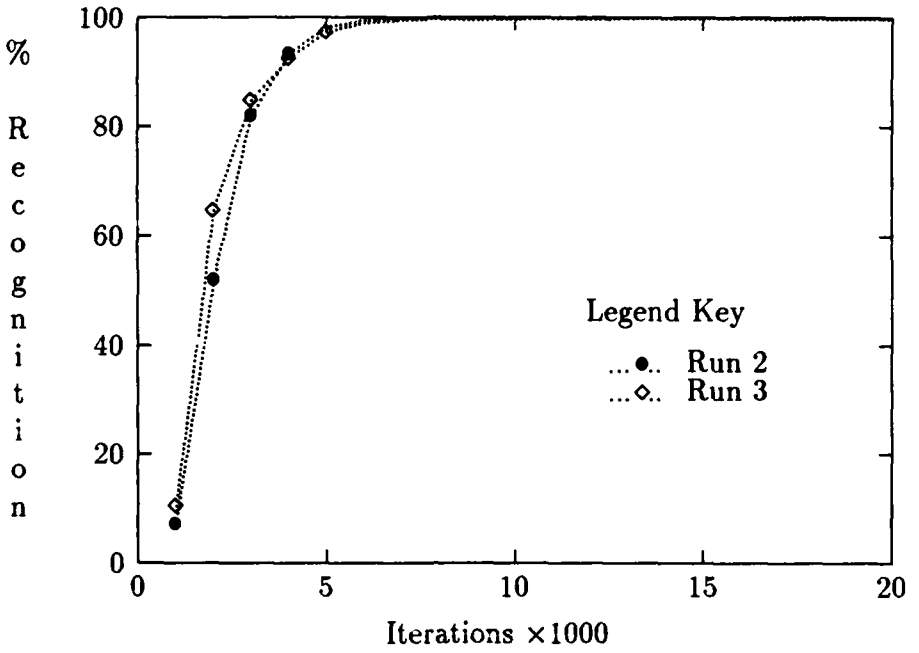

Figure 4.2. Training Histories of *Right* Classification on Training Data

by 10,000 iterations are at or near 100%. At 10,000 iterations, both runs were well past the knee of their respective curves.

4.2.3 Average Good *Classification Histories on Training Data.* The histories of average percent of *good* classification on training data for Run 2 and 3 are shown in Figure 4.3. After 1,000 iterations, the nets of Run 2 averaged close to 60%, while Run 3 nets averaged only slightly better than 50% . By 10,000 iterations, both runs were at or near 100% classification and well past the knee of their respective curves.

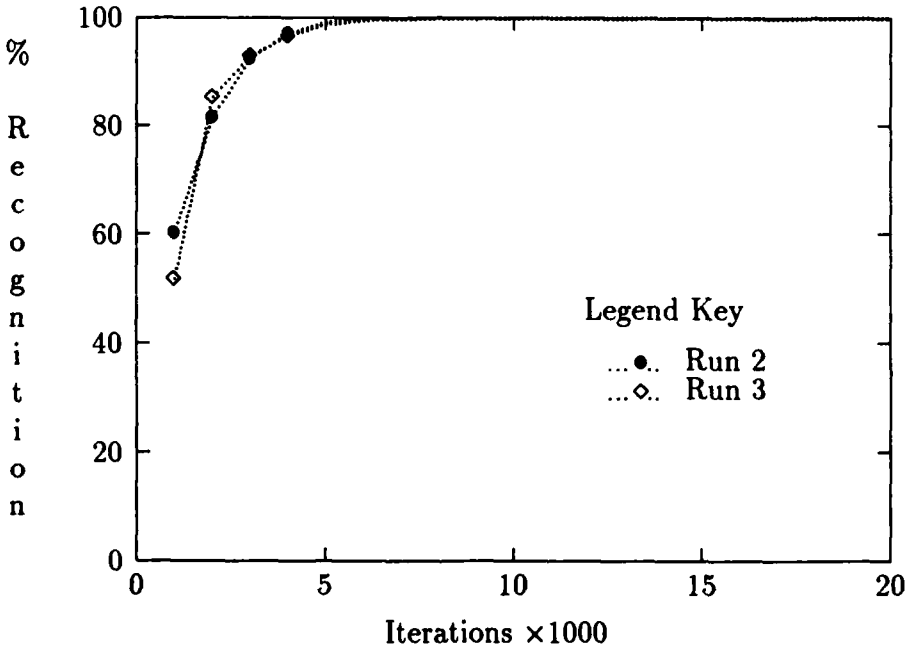

Figure 4.3. Training Histories of *Good* Classification on Training Data

4.2.4 Average Right *Classification Histories on Test Data.* Figures 4.4 and 4.5 show the histories of percent *right* classification on test data for Run 2 and Run 3, respectively. Since the performance on test data is of particular interest in this thesis, the plots show additional information about the distributions. Each figure shows plots of the average values, the average plus one standard deviation, and the average minus one standard deviation. After 1,000 iterations, the nets of Run 2 show 20% classification accuracy, while Run 3 nets were closer to 35%. Both curves are well past the knee at 10,000 iterations. At 20,000 iterations, the average for Run 3 was slightly better than for Run 2. Finally, the variance in both runs appears constant after 10,000 iterations and the variance of Run 2 is significantly greater than Run 3. At 20,000 iterations, the standard deviation was 3.597 for Run 2 and 1.892 for Run 3.

4.2.5 Average Good *Classification Histories on Test Data.* Figures 4.6 and 4.7 show the histories of percent *good* classification on test data for Run 2 and Run 3, respectively. As before, each figure has plots of the average value, the average plus one standard deviation, and the average minus one standard deviation. After 1,000 iterations, the nets of Run 2 show just below 60% classification accuracy, while Run 3 nets were closer to 70%. For both curves a point at 10,000 iterations was well past the knee. Again, after 20,000 iterations, the average for Run 3 was slightly better than for Run 2. As before, the variance in both runs appears constant after 10,000

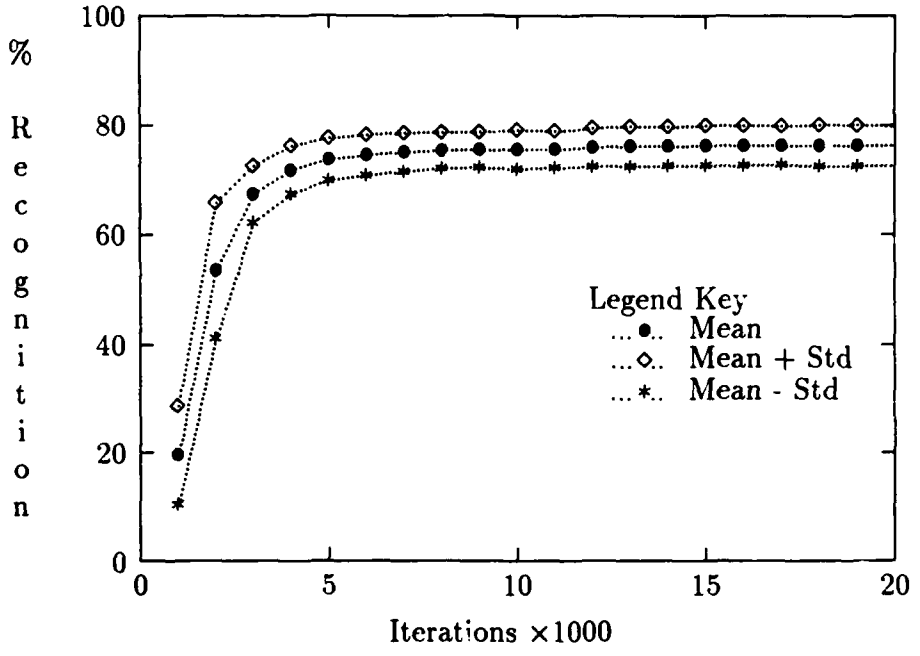

Figure 4.4. Training Histories of *Right* Classification on Test Data for Run 2

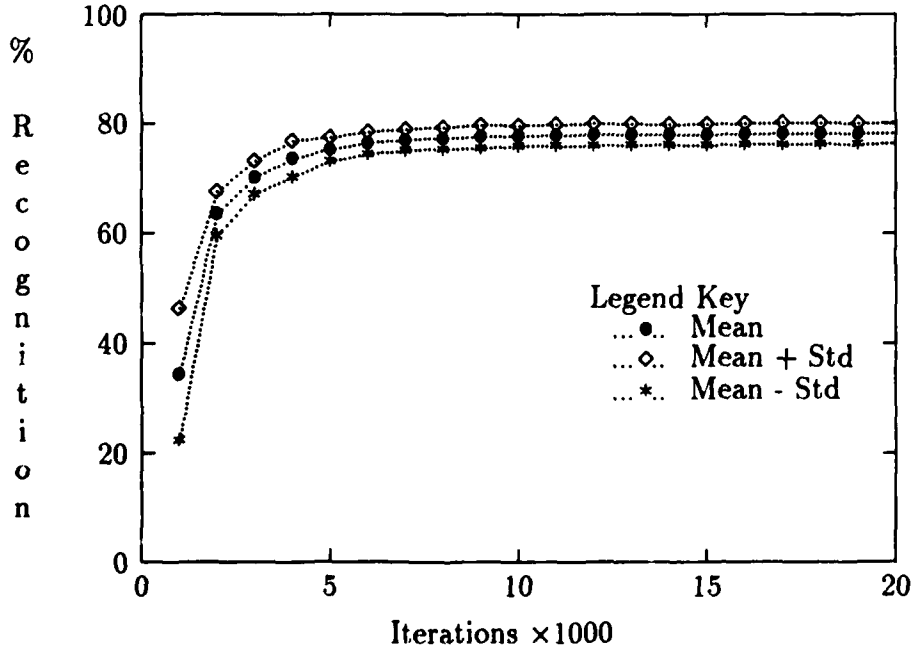

Figure 4.5. Training Histories of *Right* Classification on Test Data for Run 3

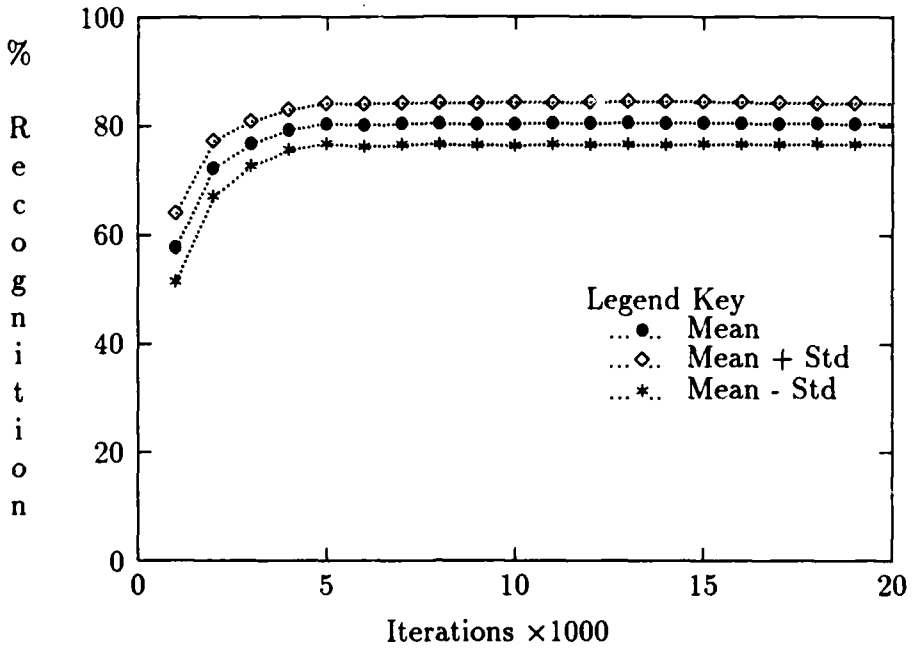

Figure 4.6. Training Histories of *Good* Classification on Test Data for Run 2

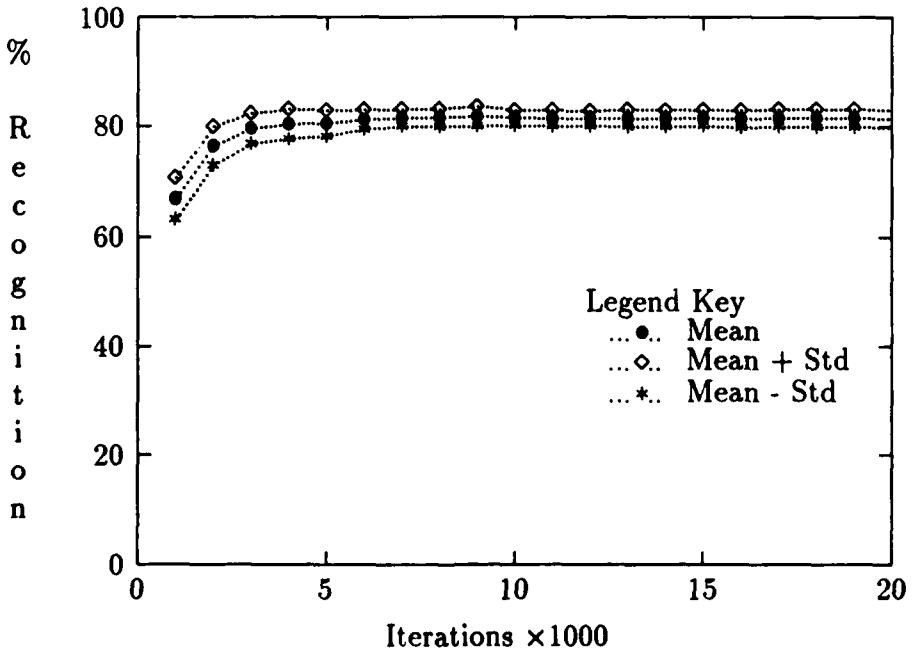

Figure 4.7. Training Histories of *Good* Classification on Test Data for Run 3

iterations and the variance of Run 2 was significantly greater than Run 3. At 20,000 iterations, the standard deviation was 3.739 for Run 2 and 1.604 for Run 3.

4.2.6 Summary of Training Performance. In general, by 10,000 iterations, the nets of both runs are asymptotically approaching their best performance. Continued training provided very little gain in any of the performance metrics. It is curious to note that even though Run 2 had better error performance than Run 3, it consistently had lower average classification performance on the test data. The algorithm for back propagation of error works by minimizing the error between the actual output and the desired output. Thus, one expects to see that nets with the smallest error yield the best classification performance. It was also observed that Run 2 networks exhibited greater variance of *good* classification on test data than did Run 3 nets. Finally, note that in both runs, the variance of *good* classification on test data was quite constant after 10,000 iterations and further training did not appear to reduce it.

4.3 Characterization of ANN Performance with the $\underline{P}$ Matrix Model.

This section presents the results of comparing the PDFs of Run 1 and Run 1A. Specifically, the conditional PDFs, $P(1 \mid 1)$ and $P(2 \mid 2)$, and the joint PDFs, P(good) of both runs, will be compared. Additionally, comparisons between the calculated P(good) PDF of Run 1 and the observed P(good) of Run 1A will be made. Recall that the calculated PDF was generated by using the observed conditional

Table 4.1. Summary Statistics for Distributions of Run 1 and Run 1A

Run ID	Metric	$P(1 \mid 1)$	$P(2 \mid 2)$	P(good)	
				Observed	Calculated
R1	Mean	0.7380	0.9213	0.8297	0.8480
	STD	0.0384	0.0185	0.0190	0.0164
R1A	Mean	0.7263	0.9304	0.8488	NA
	STD	0.0542	0.0178	0.0223	NA
Difference of Means		R1A - R1			R1A - CR1
		-0.0117	0.0091	0.0191	.0008

PDFs of the nets of Run 1 and assuming a priori probabilities of the test set exemplar classes to be $P(1) = 0.40$ and $P(2) = 0.60$.

Table 4.1 shows the average and standard deviation of the PDFs of interest. Additionally, the difference in the average values are shown in the last row of the table. The summary statistics are excerpts from Tables A.1 and A.2 located in Appendix A. The table shows that the largest difference is between the observed joint PDFs, P(good), of Run 1A and Run 1, while the smallest difference was between the observed P(good) PDF of Run 1A and the calculated P(good) PDF of Run 1.

The results of testing each PDF against the null hypothesis that they are normally distributed are shown in Table 4.2. It can be seen from the table that, at $\alpha = 0.05$, the null hypothesis must be rejected for all but two of the distributions. Based on these results, we are restricted to non-parametric tests of significance.

Table 4.3 shows the results of testing the null hypothesis that the sample distributions being compared are actually samples from the same population. Note that

Table 4.2. Results of Test for Normality for Run 1 and Run 1a Distributions

H_0 : Samples are from a normal distribution Criterion : Reject null if WS < 0.927, otherwise, Fail to reject							
Distribution	R1P11	R1P22	R1PG	R1AP11	R1A22	R1APG	CR1APG
Wilk-Shapiro Statistic	.9140	.8773	.9000	.9093	.8562	.9588	.9367
Decision	R	R	R	R	R	F	F

Table 4.3. Results of Hypothesis Tests Between Run 1 and Run 1a Distributions

H_0 : Samples are from the same distribution Criterion : Reject null if $p < 0.05$, otherwise, Fail to reject								
Test	Wilcoxon Signed Rank		Rank Sum		Kruskal-Wallis One Way AOV		Median Test	
Comparison	p	F/R	p	F/R	p	F/R	p	F/R
R1P11 - R1AP11	.1156	F	.4643	F	.4577	F	.2274	F
R1P22 - R1AP22	.0000	R	.0215	R	.0186	R	.0503	F
R1PG - R1APG	.0000	R	.0001	R	.0001	R	.0001	R
CR1PG - R1APG	.6343	F	.7731	F	.7668	F	.8172	F

there is a failure to reject the hypothesis for the comparisons between the two $P(1 \mid 1)$ PDFs and between the calculated P(good) of Run 1 and the observed P(good) of Run 1A. There is also a unanimous agreement between the tests to strongly reject the hypothesis for Run 1A and Run 1 P(good) distributions. Unfortunately, there is a conflict in the decision about the comparison of the $P(2 \mid 2)$ PDFs of the two runs. Three of the four tests dictate a rejection of the null hypothesis. It was expected that there would be a failure to reject the null hypothesis in this comparison.

The results presented in this section show no distinguishable difference between the conditional $P(1 \mid 1)$ distributions of the two runs or the observed joint distribution of Run 1A and the calculated joint distribution of Run 1. The tests show strong evidence that the two $P(2 \mid 2)$ PDFs are from different populations, which is exactly opposite of what was expected, even though these PDFs had one of the smallest differences between their averages. However, the sum of their standard deviations was also the smallest. When the variances of two samples are small, then minute differences in their means can be detected by statistical tests.

4.4 Controlling $\underline{P}$ Matrix Symmetry.

In this section, the PDFs of Run 1 and Run 2S will be compared. The summary statistics and results of hypothesis testing will be shown. Both runs were tested with identical test data sets having a 50/50% mix of class 1 and class 2 exemplars, but the training data sets for the two runs were different. The nets of Run 1 were trained with a 50/50% mix of exemplars and the nets of Run 2S were trained on a 60/40% mix of class 1 to class 2 exemplars.

In Table 4.4, the averages and standard deviations of the PDFs of Run 1 and Run 2S are shown. These figures are excerpts from tables A.1 and A.3 located in Appendix A. The last row of the table shows the difference in the average values of the like PDFs of each run. In all three instances, the differences are fairly large. The reader should note that the $\underline{P}$ matrix of Run 1 is skewed in favor of the $P(2 \mid 2)$

Table 4.4. Summary Statistics for Distributions of Run 1 and Run 2S

Run ID	Metric	$P(1 \mid 1)$	$P(2 \mid 2)$	P(good)
R1	Mean	0.7380	0.9213	0.8297
	STD	0.0384	0.0185	0.0190
R2S	Mean	0.7767	0.8320	0.8043
	STD	0.0485	0.0560	0.0374
Difference of Means		R2S - R1		
		0.0387	-0.0893	-0.0254

Table 4.5. Results of Test for Normality for Run 1 and Run 2S Distributions

H_0 : Samples are from a normal distribution Criterion : Reject null if WS < 0.927, otherwise, Fail to reject						
Distribution	R1P11	R1P22	R1PG	R2SP11	R2S22	R2SPG
Wilke-Shapiro Statistic	.9140	.8773	.9000	.9455	.9118	.9777
Decision	R	R	R	F	R	F

PDF while in Run 2S the averages of $P(1 \mid 1)$ and $P(2 \mid 2)$ PDFs are closer together and the average matrix is almost symmetric.

The results of testing each of the PDFs of interest against the null hypothesis that they are samples from a normal distribution are shown in Table 4.5. For $\alpha = 0.05$, the null hypothesis is rejected for all but two of the distributions. Since we will compare the like PDFs of the two runs, we restrict ourselves to non-parametric hypothesis tests. Table 4.6 shows the results of the these non-parametric tests. The null hypothesis assumes the sample distributions being compared come from the same population. In every case, there is unanimous agreement for strong rejection of this hypothesis.

Table 4.6. Results of Hypothesis Tests Between Run 1 and Run 2S Distributions

H_0 : Samples are from the same distribution Criterion : Reject null if $p < 0.05$, otherwise, Fail to reject								
Test	Wilcoxon Signed Rank		Rank Sum		Kruskal-Wallis One Way AOV		Median Test	
Comparison	p	F/R	p	F/R	p	F/R	p	F/R
R1P11 - R2SP11	.0081	R	.0089	R	.0078	R	.0153	R
R1P22 - R2SP22	.0000	R	.0000	R	.0000	R	.0000	R
R1PG - R2SPG	.0059	R	.0023	R	.0021	R	.0001	R

In summary, the results of this section show that all of the PDFs compared have relatively large differences in their averages. In addition, at a confidence level of $\alpha = 0.05$, the differences are significant. The tests indicate that it is highly unlikely that any of the distributions compared actually came from the same populations.

4.5 Improvement of Classification Performance via Majority Vote Rule.

This section compares the joint P(good) PDFs of runs 2 through 5. Specifically, the PDFs within each run for the single nets, the observed majority vote nets, and the calculated majority vote nets will be compared. The calculated majority vote PDFs represent the expected joint PDF of the outcomes if one assumes independence between single net outputs. The averages and results of hypothesis testing for these PDFs will be shown. All runs were tested with identical test data sets having a 50/50% mix of class 1 and class 2 exemplars. However, the training data sets for Run 2 were different than the training data for runs 3, 4, and 5. Additionally, the

Table 4.7. Summary of Means for $P($ good $)$ Distributions of Runs 2 through 5

Run	PDF ID			Differences	
	S	M	CM	M - S	CM - S
R2	0.8043	0.8223	0.8957	0.0180	0.0914
R3	0.8140	0.8200	0.8998	0.0060	0.0858
R4	0.8183	0.8190	0.8910	0.0007	0.0727
R5	0.8103	0.8083	0.8863	-0.0020	0.0760

reader is reminded that the random number generator seeds for runs 3, 4, and 5 were controlled parameters.

In Table 4.7, the averages of the P(good) PDFs generated from the runs are shown. All of the figures are excerpts from tables A.3 through A.14 located in Appendix A. The first column indicates the run to which the row applies. For the other column headers, S stands for single nets, M for observed majority vote, and CM for calculated majority vote.

In all cases, the differences between means of the calculated majority vote PDFs and the single net PDFs was substantial. These differences should be viewed as the maximum expected improvement. Thus, if we calculate $(M - S/CM - S) \times 100$ we have a measure of how close to this maximum the observed values were. These values are approximately 19.7%, 7.0%, 1%, and -2.6% for runs 2 through 5, respectively.

The results of testing each PDF for normality are shown in Table 4.8. As shown in the table we reject the null hypothesis for only the majority vote constructs of runs 4 and 5. Thus, we restrict ourselves to non-parametric tests for these runs. However, using parametric tests for runs 2 and 3 is justified. Table 4.9 shows the results of

Table 4.8. Results of Test for Normality for Distributions of Runs 2 through 5

H_0 : Samples are from a normal distribution Criterion : Reject null if WS < 0.927, otherwise, Fail to reject			
Distribution	R2SPG	R2MPG	CR2MPG
Wilke-Shapiro Statistic	.9777	.9511	.9662
Decision	F	F	F
Distribution	R3SPG	R3MPG	CR3MPG
Wilke-Shapiro Statistic	.9598	.9352	.9853
Decision	F	F	F
Distribution	R4SPG	R4MPG	CR4MPG
Wilke-Shapiro Statistic	.9755	.9155	.8787
Decision	F	R	R
Distribution	R5SPG	R5MPG	CR5MPG
Wilke-Shapiro Statistic	.9616	.8744	.9174
Decision	F	R	R

the tests of significance tests for comparisons between the PDFs of singles nets and majority vote nets within each run. The null hypothesis for all tests was equality. In every case, there is unanimous agreement for strong rejection of the hypothesis for the comparison between Run 2 single nets and Run 2 observed majority vote nets. For all others runs, there is unanimous agreement to fail to reject the hypothesis. Although not shown in the table, the same tests were performed for the differences between the PDFs of single nets and calculated majority vote nets. They were not included in the Table 4.9 because the results can be stated rather simply; out to four decimal places, all of the p values were zero. This same result applies to a comparison between the observed and calculated majority vote PDFs.

Table 4.9. Results of Hypothesis Tests Between PDFs of Single and Majority Vote Nets for Run 2 through Run 5

H_0 : Samples are from the same distribution Criterion : Reject null if $p < 0.05$, otherwise, Fail to reject										
Test	Wilcoxon Signed Rank		Rank Sum		Kruskal-Wallis One Way AOV		Median Test		Two Sample T Test	
Comparison	p	F/R	p	F/R	p	F/R	p	F/R	p	F/R
R2SPG - R2MPG	.0490	R	.0315	R	.0297	R	.0175	R	0.0249	R
R3SPG - R3MPG	.1886	F	.2009	F	.1897	F	.1677	F	0.1789	F
R4SPG - R4MPG	.7897	F	.9058	F	.8983	F	.8841	F	NA	NA
R5SPG - R5MPG	.5531	F	.6361	F	.6207	F	.3659	F	NA	NA

In summary, the results of this section show that the difference between the means of the P(good) distributions for single nets and observed majority vote nets of Run 2 is significant. The difference amounts to approximately 19.7% of the maximum expected improvement. Conversely, the observed difference in these PDFs for all other runs is not significant.

4.6 Influence of Training Data Sets, Initial Weights, and Exemplar Presentation Order on Network Solutions.

In order to gain insight into the differences between the networks of runs 2 through 5, we present here an analysis of which test exemplars were incorrectly classified in each run. Also, the number of nets that incorrectly classified each of these exemplars is shown.

Table A.15 in Appendix A contains the complete list of incorrectly classified

exemplars along with a count of how many nets misclassified each exemplar. There were 54 exemplars on the list for Run 2, 34 for Run 3, and 30 for Runs 4 and 5. The maximum number of incorrect classifications is 30, since there were 30 nets trained in each run. The lists are ordered from highest to lowest number of error counts. Since the exemplars incorrectly classified by most, or all, 30 nets in a given run are of particular interest, Table 4.10 shows the top portion of Table A.15.

Table 4.10. Partial List of Incorrectly Classified Test Exemplars in Run 2 Through Run 5

Run 2S		Run 3S		Run 4S		Run 5S	
File	Count	File	Count	File	Count	File	Count
corr96	30	corr17	30	corr17	30	corr25	30
corr25	29	corr25	30	corr25	30	corr37	30
corr194	28	corr37	30	corr37	30	corr96	30
corr53	27	corr96	30	corr41	30	corr51	30
corr227	27	corr227	30	corr96	30	corr164	30
corr164	26	corr194	30	corr194	30	corr182	30
corr7	24	corr53	29	corr168	29	corr194	30

Note that Table 4.10 shows that several exemplars appear in all four lists; corr96, corr25, corr194. Five of the exemplars appear in two or more lists and four appear in only one list. One might suspect that these exemplars were abnormal or perhaps corrupted, and should be discarded. However, a visual inspection of the graphs of these data files failed to reveal any evidence that this was the case. A comparison of all the exemplars in the lists of Table A.15 revealed that 23 exemplars were common to all four lists.

It should be evident by now that even when nets are trained with identical training sets, as in runs 3 through 5, the solutions found by each net are not all equivalent. Even if two different nets classify 20 test exemplars incorrectly, those exemplars are not necessarily the same. All that is known is that the two solutions are equivalent with respect to classification performance. What is needed is a measure of the similarity of the decision regions formed by the networks. In the following paragraphs, a metric for this purpose is proposed.

For the purpose of illustration, assume that several nets trained for a run have all found exactly the same solution. Clearly, if the decision region boundaries are the same, each net would incorrectly classify exactly the same exemplars and the error count for each one is equal to the number of nets trained in the run. There is a maximum correlation between the errors in classification made by the nets. Define N as the number of networks trained in the run, and E as the number of different exemplars on the error list. Let C_i be the the error count for the ith exemplar on the list. It should be clear that, in this case

$$\sum_{i=1}^{E} C_i/NE = 1 \tag{4.1}$$

Now assume the other extreme. Assume each of the N nets misclassifies only one test set exemplar, but each net misses a different exemplar. The error list now

contains $E = N$ exemplars, each having an error count of 1. For this situation,

$$\sum_{i=1}^{E} C_i/NE = 1/N \tag{4.2}$$

This suggests a metric, let it be called L, of the form

$$L = \sum_{i=1}^{E} C_i/NE \tag{4.3}$$

which is bounded by 1, for maximum correlation in the error list, and $1/N$, for minimum correlation. Note that if N approaches infinity, then in the case of minimum correlation, L could approach zero.

Applying this metric to the lists in Table A.15 the values of L are approximately 0.362, 0.546, 0.606, and 0.667 for runs 2, 3, 4, and 5, respectively. Since, L is a measure of the similarity of the decision regions formed by the nets of a given run, one would expect runs with lower L values to yield the greater classification improvement under a majority vote rule than runs with higher L values. Table 4.9 shows that this is true, with the exception of Run 5.

Figures 4.8 through 4.11 are graphical representations of the counts in Table A.15. Although the graphs do not show the exemplar labels, there is a one to one correspondence between the columns of the respective graphs and the entries in Table A.15 (i.e. column 1 of Figure 4.8 is the value of the count for the first entry in the list for Run 2).

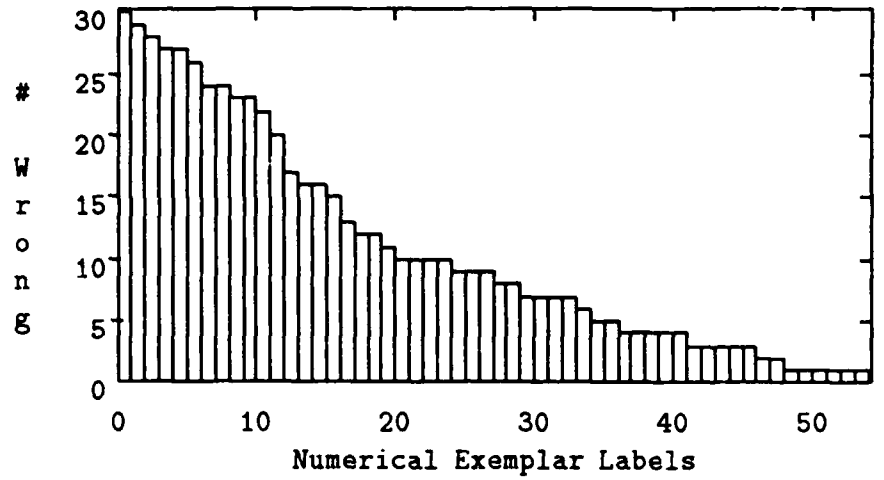

Figure 4.8. Run 2 Incorrectly Classified Exemplar Counts

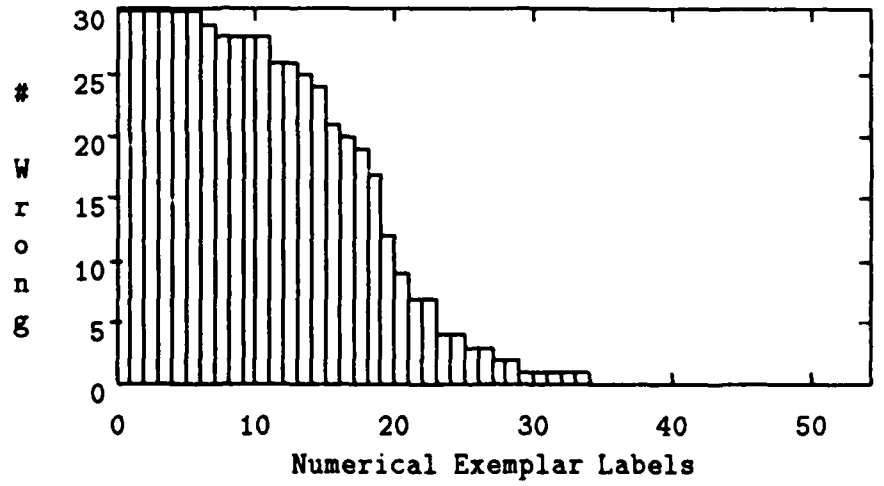

Figure 4.9. Run 3 Incorrectly Classified Exemplar Counts

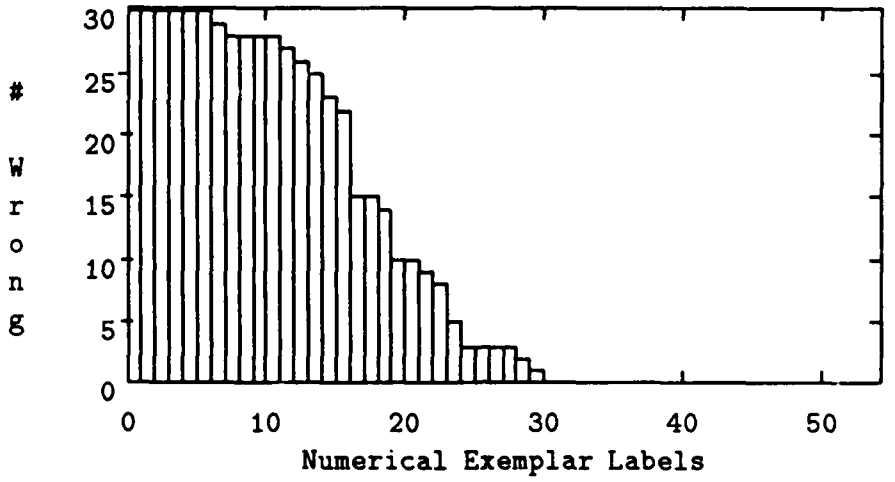

Figure 4.10. Run 4 Incorrectly Classified Exemplar Counts

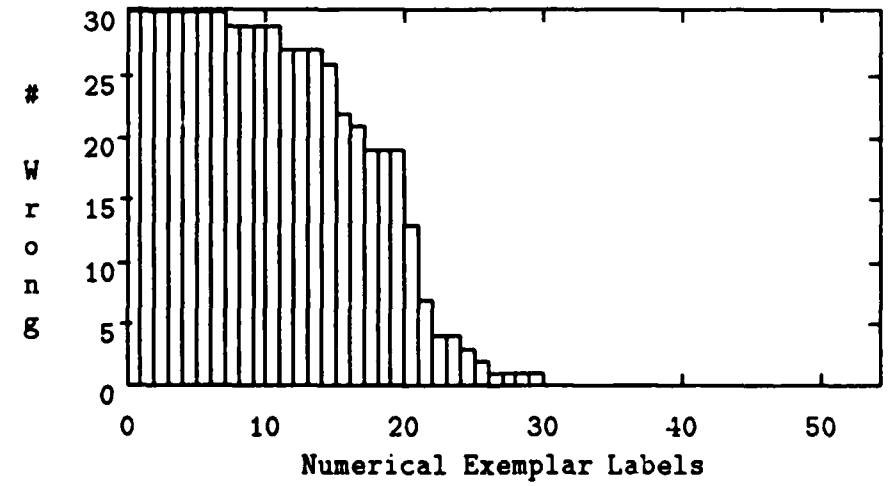

Figure 4.11. Run 5 Incorrectly Classified Exemplar Counts

The figures show, in an intuitive way, the reasons for the results presented in the previous section. Clearly, if one arbitrarily chooses any three nets from a given run, the exemplars having error counts lower than 15 (i.e. less than a 0.5 probability of being incorrectly classified in that run) have a greater than 0.5 probability of being correctly classified under a majority vote rule taken over the three nets. Observe in Figures 4.8, that Run 2 has 38 exemplars, approximately 70%, which fall into this category. For Runs 3, 4, and 5, the proportions of exemplars in this category are approximately 44%, 40%, and 33%, respectively. Based on these proportions, it could be predicted, for a majority vote rule over three nets, that Run 2 nets would yield the greatest improvement in classification performance, Run 3 the next greatest, Run 4 the next, and finally Run 5. The results presented in Table 4.7 support this prediction.

4.7 Summary.

This chapter covered the major results of this thesis effort. First, the average training performance histories, for a variety of metrics, were presented. This provided information about how fast the networks trained, the general shape and variances of the curves, and the relationships between the performance metrics. Next, evidence demonstrating the stationarity and control of $\underline{P}$ matrix distributions constructed from network responses to test data was examined. Then, results of comparisons between the performance PDFs of single networks and majority vote networks for four

different training runs, were examined. Finally, a close inspection of the incorrectly classified exemplars for each run was performed.

V. Conclusions and Recommendations

This final chapter contains the closing remarks and conclusions based on the evidence and results presented in Chapter IV. In addition, recommendations for future research are provided. The reader should be aware that the conclusions presented here only apply to the two class recognition problem for the data sets used in this study. It would be premature to apply these conclusions to ANNs trained on other data sets or even a three class problem using the same type of data sets.

5.1 Conclusions.

5.1.1 Training Performance.

Conclusion. A three-layer back propagation neural network can train directly on correlation signatures of direct sequence (DS) and linearly stepped frequency hopped (FH) spread spectrum signals. The networks can be expected to yield at or near 100% classification accuracy on training data sets and at or near 80% accuracy on test data sets after 10,000 training iterations. Additionally, the overall standard deviation of test set classification accuracy can be expected to be less than 2%.

Discussion. It is clear from the plots shown in Section 4.2 that by 10,000 iterations, the nets were at or near their maximum values for any of

the classification performance metrics. Additional training past 10,000 iterations provided little, if any, improvement in test set classification accuracy. Additionally, no reduction in the variance was observed.

5.1.2 Characterization of ANN Performance with the $\underline{P}$ Matrix Model.

Conclusion. The $\underline{P}$ matrix model is a valid and useful tool for describing and evaluating ANN classification performance.

Discussion. It was shown in Section 4.3 that the $\underline{P}$ matrix obtained from testing 30 networks on one test data set accurately predicted the joint PDF obtained by testing the nets with data sets having a different proportions of exemplars. There was essentially no difference in the $\underline{P}$ matrices obtained from testing the nets with either test data set. While this statement is true for the conditional PDFs of class 1 exemplars, it may be debatable for the PDFs of class 2 exemplars. However, it was observed that the nets did very well at recognizing class 2 exemplars and the PDFs had much smaller variances. This small variance caused three of four tests of significance to reject the null hypothesis that the distributions were the same, even though there was a relatively small difference in their means. Thus, it may be argued, from a practical viewpoint, that there was actually no difference between these PDFs. The implication of these observations is that the responses of the networks were stationary. The change observed in the average joint $P(\text{good})$ PDF was due only to the change in the a priori probabilities $P(1)$ and $P(2)$. Thus,

the conditional $\underline{P}$ matrix is a more useful description of a network's classification performance than the joint probability, P(good).

5.1.3 Controlling $\underline{P}$ Matrix Symmetry.

Conclusion. Given that training on a particular data set does not yield the desired conditional classification performances, it is possible to change this by appropriate adjustments in the proportions of exemplar classes in the training data set.

Discussion. The results in Section 4.4 showed the average conditional probabilities, $P(1 \mid 1)$ and $P(2 \mid 2)$, for a particular run of 30 nets to be 0.7380 and 0.9213. The average joint recognition probability was 0.8297. Obviously, These nets were not recognizing class 1 exemplars as well as class 2 exemplars. The training data set proportions were changed from a 50/50% mix to a 60/40% mix of class 1 to class 2 exemplars and another run of 30 nets were trained. The resulting average conditional probabilities, $P(1 \mid 1)$ and $P(2 \mid 2)$ were 0.7767 and 0.8320, while the joint probability was 0.8043. Statistical tests on the PDFs confirmed that the changes were significant. The $\underline{P}$ matrix was indeed adjusted toward symmetry. Clearly, this demonstrates a technique for controlling the symmetry of ANN classification behavior. However, the trade-off was a reduction in the joint classification accuracy. This result is not too surprising because the back propagation of error algorithm seeks to minimize the error over *all* training exemplars. If the training

data set is weighted heavily toward on exemplar class, then clearly, the solution in the weight space will shift to favor that class since it now has a greater contribution to the overall error. Of course, the exact proportions needed to achieve symmetry would be dependent on the relative contribution to the overall error of each class of exemplars taken as a group. While the experimental objective was to adjust the $\underline{P}$ matrix toward symmetry, this may not be desirable for all cases. The point is that a network's classification response can be adjusted to whatever symmetry is best for a given application.

5.1.4 Improvement of Classification Performance via the Majority Vote Rule.

Conclusion. If one trains three networks with three different, but equivalent, training data sets, it is possible to use a majority vote rule to realize an improvement in average classification performance.

Discussion. In Section 4.5, it was shown that 30 majority vote networks, constructed from individual nets trained on slightly different data sets, averaged 1.8% better performance than 30 individual nets trained in the same manner. Statistical tests on the joint recognition PDFs showed this difference to be significant. The differences in majority vote net performance and individual net performance for three other runs of 30 nets trained on identical training data sets were not significant.

5.1.5 Influence of Training Data Sets, Initial Weights, and Exemplar Presentation Order on Network Solutions.

Conclusion. The influence of the training data set strongly outweighs the influence of initial starting weights and/or the order of presentation of training exemplars, with regard to the decision regions formed by a given network.

Discussion. In Section 4.6, an examination of the incorrectly classified test set exemplars was performed. A set of 30 nets were trained for each of four runs. In Run 2, the exact composition of the training set, initial starting weights, and presentation order of exemplars, was different for each net trained. In Run 3, only the initial weights and presentation order were different. In Run 4, only the initial weights were different and in Run 5, only the presentation orders were different. The values of L, a metric measuring the correlation between the decision regions formed by the nets within a given run, for Runs 2, 3, 4, and 5 were 0.362, 0.546, 0.606, and 0.667, respectively. These values clearly suggest a relative order and degree of impact the conditions have on the exact shape of decision regions formed. The L values indicate that exact composition of the training data set is most important, the initial set of weights the next most important, and the order of exemplar presentation the least important.

5.2 *Recommendations.*

1. The validity of the findings in this thesis should be tested against more complex recognition problems. This could be done by obtaining samples of the correlation signatures of randomly driven FH and FH/DS signals, and repeating the experiments performed in this study for a four class recognition problem.

2. For future research involving this application of ANNs, an appropriate amount of white gaussian noise should be added to the simulated spread spectrum signals before the correlation signatures are obtained. The classification performance of ANNs in noisy environments could then be explored.

3. Investigations should be made to determine if ANNs can learn to classify correlation signatures according to some other parameter, such as chip rate or code length.

4. Test data sets used to evaluate ANN classification performance should be composed of a uniform mix of at least 25 exemplars for each exemplar class. Clearly, the test data set is a measuring device for determining generalized classification capabilities of trained networks. As a rule of thumb, if it is desired to measure a performance metric on a given trial to the nearest $\pm 1/2$ unit of the metric, then the measuring device should have a tolerance of ± 1 unit of measure. Additionally, a uniform mix of exemplar classes will provide an unbiased estimate of classification accuracy. It was shown in this thesis, that if the $\underline{P}$ matrix for a trained network is not symmetric, it is possible to shift the value of the joint

classification performance significantly simply by biasing the test set in favor of one exemplar class or the other. Future research involving evaluation of ANN performance should report conditional classification accuracies, as well as the overall joint accuracy.

Appendix A. *Data Tables*

The following data tables were contructed from results of network training runs. Networks were trained to 20,000 iterations and performance tested against a test data set. Exemplars in the test data sets were not included in the training data sets. Each observed majority vote network was constructed by drawing three independently trained single nets from a pool of 90 without replacement. These nets would vote on the final classification of each test exemplar. Each calculated majority vote network was contructed by using the $\underline{P}$ matrices constructed for the three selected nets. This calculated matrix assumes statistical independence between the outcome of individual nets, which is to say that there is no relationship between the classification outcome of any two nets for a particular test exemplar. Although only the $P(1 \mid 1)$, $P(2 \mid 2)$, and $P(\text{good})$ distributions are used, the $P(2 \mid 1)$ and $P(1 \mid 2)$ probabilities are shown for completeness.

Table A.1. Observed Probability Matrices for Run 1

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good) Observed	P(good) Calculated
net1	0.74	0.26	0.08	0.92	0.83	0.848
net2	0.74	0.26	0.08	0.92	0.83	0.848
net3	0.78	0.22	0.08	0.92	0.85	0.864
net4	0.74	0.26	0.08	0.92	0.83	0.848
net5	0.72	0.28	0.08	0.92	0.82	0.840
net6	0.76	0.24	0.04	0.96	0.86	0.880
net7	0.72	0.28	0.06	0.94	0.83	0.852
net8	0.74	0.26	0.08	0.92	0.83	0.848
net9	0.80	0.20	0.10	0.90	0.85	0.860
net10	0.66	0.34	0.08	0.92	0.79	0.816
net11	0.72	0.28	0.08	0.92	0.82	0.840
net12	0.74	0.26	0.06	0.94	0.84	0.860
net13	0.76	0.24	0.04	0.96	0.86	0.880
net14	0.76	0.24	0.10	0.90	0.83	0.844
net15	0.80	0.20	0.12	0.88	0.84	0.848
net16	0.76	0.24	0.08	0.92	0.84	0.856
net17	0.76	0.24	0.12	0.88	0.82	0.832
net18	0.68	0.32	0.08	0.92	0.80	0.824
net19	0.78	0.22	0.10	0.90	0.84	0.852
net20	0.72	0.28	0.08	0.92	0.82	0.840
net21	0.78	0.22	0.08	0.92	0.85	0.864
net22	0.76	0.24	0.08	0.92	0.84	0.856
net23	0.74	0.26	0.08	0.92	0.83	0.848
net24	0.74	0.26	0.06	0.94	0.84	0.860
net25	0.64	0.36	0.06	0.94	0.79	0.820
net26	0.74	0.26	0.06	0.94	0.84	0.860
net27	0.66	0.34	0.10	0.90	0.78	0.804
net28	0.76	0.24	0.08	0.92	0.84	0.856
net29	0.74	0.26	0.08	0.92	0.83	0.848
net30	0.70	0.30	0.06	0.94	0.82	0.844
Mean	0.7380	0.2620	0.0787	0.9213	0.8297	0.8480
STD	0.0384	0.0384	0.0185	0.0185	0.0190	0.0164

Table A.2. Observed Probability Matrices for Run 1a

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{ good })$
net1_a	0.763	0.237	0.070	0.930	0.863
net2_a	0.711	0.289	0.070	0.930	0.842
net3_a	0.816	0.184	0.070	0.930	0.884
net4_a	0.711	0.289	0.070	0.930	0.842
net5_a	0.711	0.289	0.070	0.930	0.842
net6_a	0.789	0.211	0.035	0.965	0.895
net7_a	0.763	0.237	0.053	0.947	0.874
net8_a	0.763	0.237	0.070	0.930	0.863
net9_a	0.789	0.211	0.088	0.912	0.863
net10_a	0.658	0.342	0.070	0.930	0.821
net11_a	0.658	0.342	0.070	0.930	0.821
net12_a	0.711	0.289	0.053	0.947	0.853
net13_a	0.763	0.237	0.035	0.965	0.884
net14_a	0.711	0.289	0.088	0.912	0.832
net15_a	0.739	0.211	0.123	0.877	0.842
net16_a	0.737	0.263	0.070	0.930	0.853
net17_a	0.737	0.263	0.105	0.895	0.832
net18_a	0.711	0.289	0.070	0.930	0.842
net19_a	0.789	0.211	0.088	0.912	0.863
net20_a	0.658	0.342	0.070	0.930	0.821
net21_a	0.763	0.237	0.070	0.930	0.863
net22_a	0.737	0.263	0.070	0.930	0.853
net23_a	0.737	0.263	0.070	0.930	0.853
net24_a	0.684	0.316	0.053	0.947	0.842
net25_a	0.553	0.447	0.053	0.947	0.789
net26_a	0.684	0.316	0.053	0.947	0.842
net27_a	0.658	0.342	0.088	0.912	0.811
net28_a	0.763	0.237	0.070	0.930	0.863
net29_a	0.763	0.237	0.070	0.930	0.863
net30_a	0.711	0.289	0.053	0.947	0.853
Mean	0.7263	0.2737	0.0696	0.9304	0.8488
STD	0.0542	0.0542	0.0178	0.0178	0.0223

Table A.3. Observed Probability Matrices for Run 2 Single Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.82	0.18	0.30	0.70	0.76
net2	0.86	0.14	0.20	0.80	0.83
net3	0.84	0.16	0.22	0.78	0.81
net4	0.78	0.22	0.16	0.84	0.81
net5	0.74	0.26	0.12	0.88	0.81
net6	0.74	0.26	0.24	0.76	0.75
net7	0.74	0.26	0.10	0.90	0.82
net8	0.78	0.22	0.20	0.80	0.79
net9	0.76	0.24	0.12	0.88	0.82
net10	0.82	0.18	0.18	0.82	0.82
net11	0.76	0.24	0.10	0.90	0.83
net12	0.76	0.24	0.12	0.88	0.82
net13	0.72	0.28	0.22	0.78	0.75
net14	0.74	0.26	0.22	0.78	0.76
net15	0.76	0.24	0.16	0.84	0.80
net16	0.82	0.18	0.14	0.86	0.84
net17	0.74	0.26	0.18	0.82	0.78
net18	0.80	0.20	0.22	0.78	0.79
net19	0.70	0.30	0.18	0.82	0.76
net20	0.80	0.20	0.12	0.88	0.84
net21	0.78	0.22	0.16	0.84	0.81
net22	0.82	0.18	0.10	0.90	0.86
net23	0.72	0.28	0.12	0.88	0.80
net24	0.72	0.28	0.12	0.88	0.80
net25	0.74	0.26	0.32	0.68	0.71
net26	0.72	0.28	0.18	0.82	0.77
net27	0.80	0.20	0.10	0.90	0.85
net28	0.90	0.10	0.14	0.86	0.88
net29	0.76	0.24	0.16	0.84	0.80
net30	0.86	0.14	0.14	0.86	0.86
Mean	0.7767	0.2233	0.1680	0.8320	0.8043
STD	0.0485	0.0485	0.0560	0.0560	0.0374

Table A.4. Observed Probability Matrices for Run 2 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid i)$	$P(i \mid 2)$	$P(2 \mid 2)$	P(good)
mvnet1	0.84	0.16	0.20	0.80	0.82
mvnet2	0.76	0.24	0.10	0.90	0.83
mvnet3	0.80	0.20	0.14	0.86	0.83
mvnet4	0.82	0.18	0.16	0.84	0.83
mvnet5	0.78	0.22	0.14	0.86	0.82
mvnet6	0.76	0.24	0.08	0.92	0.84
mvnet7	0.80	0.20	0.16	0.84	0.82
mvnet8	0.76	0.24	0.12	0.88	0.82
mvnet9	0.78	0.22	0.16	0.84	0.81
mvnet10	0.82	0.18	0.24	0.76	0.79
mvnet11	0.84	0.16	0.20	0.80	0.82
mvnet12	0.74	0.26	0.14	0.86	0.80
mvnet13	0.78	0.22	0.12	0.88	0.83
mvnet14	0.84	0.16	0.16	0.84	0.84
mvnet15	0.78	0.22	0.12	0.88	0.83
mvnet16	0.80	0.20	0.12	0.88	0.84
mvnet17	0.76	0.24	0.14	0.86	0.81
mvnet18	0.78	0.22	0.12	0.88	0.83
mvnet19	0.78	0.22	0.18	0.82	0.80
mvnet20	0.78	0.22	0.20	0.80	0.79
mvnet21	0.90	0.10	0.20	0.80	0.85
mvnet22	0.80	0.20	0.08	0.92	0.86
mvnet23	0.76	0.24	0.10	0.90	0.83
mvnet24	0.74	0.26	0.18	0.82	0.78
mvnet25	0.78	0.22	0.12	0.88	0.83
mvnet26	0.80	0.20	0.12	0.88	0.84
mvnet27	0.78	0.22	0.16	0.84	0.81
mvnet28	0.76	0.24	0.10	0.90	0.83
mvnet29	0.82	0.18	0.14	0.86	0.84
mvnet30	0.78	0.22	0.18	0.82	0.80
Mean	0.7907	0.2093	0.1460	0.8540	0.8223
STD	0.0338	0.0338	0.0387	0.0387	0.0184

Table A.5. Calculated Probability Matrices for Run 2 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{ good })$
cmvnet1	0.92	0.08	0.09	0.91	0.92
cmvnet2	0.91	0.09	0.10	0.90	0.91
cmvnet3	0.86	0.14	0.07	0.93	0.89
cmvnet4	0.90	0.10	0.06	0.94	0.92
cmvnet5	0.88	0.12	0.11	0.89	0.88
cmvnet6	0.86	0.14	0.05	0.95	0.91
cmvnet7	0.86	0.14	0.09	0.91	0.89
cmvnet8	0.81	0.19	0.03	0.97	0.89
cmvnet9	0.87	0.13	0.10	0.90	0.88
cmvnet10	0.90	0.10	0.13	0.87	0.88
cmvnet11	0.93	0.07	0.09	0.91	0.92
cmvnet12	0.86	0.14	0.10	0.90	0.88
cmvnet13	0.86	0.14	0.06	0.94	0.90
cmvnet14	0.90	0.10	0.07	0.93	0.91
cmvnet15	0.83	0.17	0.10	0.90	0.86
cmvnet16	0.88	0.12	0.07	0.93	0.90
cmvnet17	0.88	0.12	0.07	0.93	0.91
cmvnet18	0.86	0.14	0.04	0.96	0.91
cmvnet19	0.85	0.15	0.13	0.87	0.86
cmvnet20	0.89	0.11	0.14	0.86	0.87
cmvnet21	0.94	0.06	0.09	0.91	0.92
cmvnet22	0.90	0.10	0.07	0.93	0.92
cmvnet23	0.86	0.14	0.05	0.95	0.90
cmvnet24	0.82	0.18	0.08	0.92	0.87
cmvnet25	0.88	0.12	0.10	0.90	0.89
cmvnet26	0.87	0.13	0.05	0.95	0.91
cmvnet27	0.88	0.12	0.07	0.93	0.90
cmvnet28	0.81	0.19	0.03	0.97	0.89
cmvnet29	0.88	0.12	0.11	0.89	0.89
cmvnet30	0.88	0.12	0.11	0.89	0.88
Mean	0.8737	0.1263	0.0822	0.9178	0.8957
STD	0.0313	0.0313	0.0296	0.0296	0.0168

Table A.6. Observed Probability Matrices for Run 3 Single Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
net1	0.68	0.32	0.08	0.92	0.80
net2	0.70	0.30	0.06	0.94	0.82
net3	0.78	0.22	0.10	0.90	0.84
net4	0.70	0.30	0.04	0.96	0.83
net5	0.72	0.28	0.10	0.90	0.81
net6	0.66	0.34	0.06	0.94	0.80
net7	0.72	0.28	0.08	0.92	0.82
net8	0.72	0.28	0.08	0.92	0.82
net9	0.70	0.30	0.08	0.92	0.81
net10	0.72	0.28	0.08	0.92	0.82
net11	0.68	0.32	0.04	0.96	0.82
net12	0.74	0.26	0.08	0.92	0.83
net13	0.68	0.32	0.06	0.94	0.81
net14	0.78	0.22	0.10	0.90	0.84
net15	0.70	0.30	0.08	0.92	0.81
net16	0.70	0.30	0.10	0.90	0.80
net17	0.66	0.34	0.08	0.92	0.79
net18	0.76	0.24	0.08	0.92	0.84
net19	0.68	0.32	0.06	0.94	0.81
net20	0.70	0.30	0.12	0.88	0.79
net21	0.72	0.28	0.10	0.90	0.81
net22	0.72	0.28	0.10	0.90	0.81
net23	0.70	0.30	0.10	0.90	0.80
net24	0.70	0.30	0.06	0.94	0.82
net25	0.66	0.34	0.10	0.90	0.78
net26	0.70	0.30	0.10	0.90	0.80
net27	0.74	0.26	0.10	0.90	0.82
net28	0.76	0.24	0.06	0.94	0.85
net29	0.70	0.30	0.10	0.90	0.80
net30	0.76	0.24	0.12	0.88	0.82
Mean	0.7113	0.2887	0.0833	0.9167	0.8140
STD	0.0325	0.0325	0.0207	0.0207	0.0160

Table A.7. Observed Probability Matrices for Run 3 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
mvnet1	0.74	0.26	0.10	0.90	0.82
mvnet2	0.74	0.26	0.10	0.90	0.82
mvnet3	0.70	0.30	0.10	0.90	0.80
mvnet4	0.72	0.28	0.06	0.94	0.83
mvnet5	0.74	0.26	0.08	0.92	0.83
mvnet6	0.68	0.32	0.10	0.90	0.79
mvnet7	0.72	0.28	0.08	0.92	0.82
mvnet8	0.68	0.32	0.08	0.92	0.80
mvnet9	0.72	0.28	0.10	0.90	0.81
mvnet10	0.68	0.32	0.06	0.94	0.81
mvnet11	0.78	0.22	0.04	0.96	0.87
mvnet12	0.72	0.28	0.08	0.92	0.82
mvnet13	0.70	0.30	0.08	0.92	0.81
mvnet14	0.74	0.26	0.10	0.90	0.82
mvnet15	0.72	0.28	0.08	0.92	0.82
mvnet16	0.68	0.32	0.10	0.90	0.79
mvnet17	0.78	0.22	0.08	0.92	0.85
mvnet18	0.70	0.30	0.10	0.90	0.80
mvnet19	0.74	0.26	0.08	0.92	0.83
mvnet20	0.70	0.30	0.06	0.94	0.82
mvnet21	0.76	0.24	0.08	0.92	0.84
mvnet22	0.70	0.30	0.08	0.92	0.81
mvnet23	0.74	0.26	0.08	0.92	0.83
mvnet24	0.76	0.24	0.06	0.94	0.85
mvnet25	0.74	0.26	0.10	0.90	0.82
mvnet26	0.70	0.30	0.10	0.90	0.80
mvnet27	0.72	0.28	0.08	0.92	0.82
mvnet28	0.74	0.26	0.08	0.92	0.83
mvnet29	0.70	0.30	0.08	0.92	0.81
mvnet30	0.74	0.26	0.08	0.92	0.83
Mean	0.7227	0.2773	0.0827	0.9173	0.8200
STD	0.0277	0.0277	0.0153	0.0153	0.0175

Table A.8. Calculated Probability Matrices for Run 3 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P($ good $)$
cmvnet1	0.83	0.17	0.03	0.97	0.90
cmvnet2	0.82	0.18	0.02	0.98	0.90
cmvnet3	0.79	0.21	0.02	0.98	0.88
cmvnet4	0.82	0.18	0.01	0.99	0.90
cmvnet5	0.80	0.20	0.02	0.98	0.89
cmvnet6	0.80	0.20	0.01	0.99	0.89
cmvnet7	0.83	0.17	0.01	0.99	0.91
cmvnet8	0.78	0.22	0.02	0.98	0.88
cmvnet9	0.80	0.20	0.02	0.98	0.89
cmvnet10	0.82	0.18	0.01	0.99	0.90
cmvnet11	0.83	0.17	0.01	0.99	0.91
cmvnet12	0.82	0.18	0.02	0.98	0.90
cmvnet13	0.82	0.18	0.02	0.98	0.90
cmvnet14	0.80	0.20	0.03	0.97	0.89
cmvnet15	0.81	0.19	0.01	0.99	0.90
cmvnet16	0.83	0.17	0.02	0.98	0.90
cmvnet17	0.87	0.13	0.02	0.98	0.92
cmvnet18	0.78	0.22	0.02	0.98	0.88
cmvnet19	0.83	0.17	0.01	0.99	0.91
cmvnet20	0.84	0.16	0.01	0.99	0.92
cmvnet21	0.85	0.15	0.02	0.98	0.91
cmvnet22	0.81	0.19	0.02	0.98	0.90
cmvnet23	0.82	0.18	0.02	0.98	0.90
cmvnet24	0.84	0.16	0.01	0.99	0.91
cmvnet25	0.82	0.18	0.02	0.98	0.90
cmvnet26	0.80	0.20	0.02	0.98	0.89
cmvnet27	0.82	0.18	0.02	0.98	0.90
cmvnet28	0.82	0.18	0.02	0.98	0.90
cmvnet29	0.83	0.17	0.02	0.98	0.91
cmvnet30	0.82	0.18	0.02	0.98	0.90
Mean	0.8185	0.1815	0.0190	0.9810	0.8998
STD	0.0182	0.0182	0.0052	0.0052	0.0098

Table A.9. Observed Probability Matrices for Run 4 Single Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
net121	0.74	0.26	0.06	0.94	0.84
net122	0.68	0.32	0.08	0.92	0.80
net123	0.72	0.28	0.06	0.94	0.83
net124	0.72	0.28	0.08	0.92	0.82
net125	0.70	0.30	0.08	0.92	0.81
net126	0.76	0.24	0.06	0.94	0.85
net127	0.66	0.34	0.08	0.92	0.79
net128	0.66	0.34	0.06	0.94	0.80
net129	0.68	0.32	0.10	0.90	0.79
net130	0.64	0.36	0.08	0.92	0.78
net131	0.68	0.32	0.04	0.96	0.82
net132	0.74	0.26	0.04	0.96	0.85
net133	0.70	0.30	0.08	0.92	0.81
net134	0.72	0.28	0.08	0.92	0.82
net135	0.74	0.26	0.08	0.92	0.83
net136	0.76	0.24	0.08	0.92	0.84
net137	0.68	0.32	0.04	0.96	0.82
net138	0.72	0.28	0.06	0.94	0.83
net139	0.74	0.26	0.10	0.90	0.82
net140	0.66	0.34	0.06	0.94	0.80
net141	0.68	0.32	0.08	0.92	0.80
net142	0.68	0.32	0.06	0.94	0.81
net143	0.76	0.24	0.10	0.90	0.83
net144	0.72	0.28	0.08	0.92	0.82
net145	0.70	0.30	0.10	0.90	0.80
net146	0.72	0.28	0.08	0.92	0.82
net147	0.74	0.26	0.06	0.94	0.84
net148	0.74	0.26	0.06	0.94	0.84
net149	0.64	0.36	0.08	0.92	0.78
net150	0.78	0.22	0.06	0.94	0.86
Mean	0.7087	0.2913	0.0720	0.9280	0.8183
STD	0.0371	0.0371	0.0168	0.0168	0.0205

Table A.10. Observed Probability Matrices for Run 4 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
mvnet31	0.74	0.26	0.08	0.92	0.83
mvnet32	0.74	0.26	0.08	0.92	0.83
mvnet33	0.68	0.32	0.08	0.92	0.80
mvnet34	0.72	0.28	0.08	0.92	0.82
mvnet35	0.70	0.30	0.06	0.94	0.82
mvnet36	0.72	0.28	0.08	0.92	0.82
mvnet37	0.74	0.26	0.08	0.92	0.83
mvnet38	0.68	0.32	0.06	0.94	0.81
mvnet39	0.74	0.26	0.10	0.90	0.82
mvnet40	0.66	0.34	0.06	0.94	0.80
mvnet41	0.74	0.26	0.06	0.94	0.84
mvnet42	0.72	0.28	0.04	0.96	0.84
mvnet43	0.70	0.30	0.08	0.92	0.81
mvnet44	0.72	0.28	0.10	0.90	0.81
mvnet45	0.74	0.26	0.10	0.90	0.82
mvnet46	0.74	0.26	0.08	0.92	0.83
mvnet47	0.70	0.30	0.10	0.90	0.80
mvnet48	0.70	0.30	0.04	0.96	0.83
mvnet49	0.72	0.28	0.08	0.92	0.82
mvnet50	0.72	0.28	0.06	0.94	0.83
mvnet51	0.74	0.26	0.06	0.94	0.84
mvnet52	0.66	0.34	0.06	0.94	0.80
mvnet53	0.72	0.28	0.08	0.92	0.82
mvnet54	0.68	0.32	0 06	0.94	0.81
mvnet55	0.76	0.24	0.10	0.90	0.83
mvnet56	0.74	0.26	0.08	0.92	0.83
mvnet57	0.68	0.32	0.08	0.92	0.80
mvnet58	0.70	0.30	0.08	0.92	0.81
mvnet59	0.72	0.28	0.08	0.92	0.82
mvnet60	0.70	0.30	0.10	0.90	0.80
Mean	0.7140	0.2860	0.0760	0.9240	0.8190
STD	0.0259	0.0259	0.0167	0.0167	0.0127

Table A.11. Calculated Probability Matrices for Run 4 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P($ good $)$
cmvnet31	0.83	0.17	0.02	0.98	0.91
cmvnet32	0.84	0.16	0.02	0.98	0.91
cmvnet33	0.78	0.22	0.02	0.98	0.88
cmvnet34	0.78	0.22	0.02	0.98	0.88
cmvnet35	0.78	0.22	0.01	0.99	0.88
cmvnet36	0.81	0.19	0.02	0.98	0.89
cmvnet37	0.83	0.17	0.02	0.98	0.91
cmvnet38	0.78	0.22	0.01	0.99	0.89
cmvnet39	0.82	0.18	0.03	0.97	0.89
cmvnet40	0.77	0.23	0.02	0.98	0.88
cmvnet41	0.83	0.17	0.01	0.99	0.91
cmvnet42	0.80	0.20	0.01	0.99	0.90
cmvnet43	0.78	0.22	0.02	0.98	0.88
cmvnet44	0.79	0.21	0.02	0.98	0.89
cmvnet45	0.82	0.18	0.02	0.98	0.90
cmvnet46	0.84	0.16	0.02	0.98	0.91
cmvnet47	0.82	0.18	0.02	0.98	0.90
cmvnet48	0.79	0.21	0.02	0.98	0.89
cmvnet49	0.78	0.22	0.02	0.98	0.88
cmvnet50	0.79	0.21	0.02	0.98	0.89
cmvnet51	0.80	0.20	0.02	0.98	0.89
cmvnet52	0.76	0.24	0.01	0.99	0.87
cmvnet53	0.81	0.19	0.02	0.98	0.90
cmvnet54	0.77	0.23	0.01	0.99	0.88
cmvnet55	0.83	0.17	0.02	0.98	0.91
cmvnet56	0.82	0.18	0.02	0.98	0.90
cmvnet57	0.78	0.22	0.02	0.98	0.88
cmvnet58	0.78	0.22	0.02	0.98	0.88
cmvnet59	0.78	0.22	0.02	0.98	0.88
cmvnet60	0.78	0.22	0.02	0.98	0.88
Mean	0.7992	0.2008	0.0171	0.9829	0.8910
STD	0.0228	0.0228	0.0041	0.0041	0.0112

Table A.12. Observed Probability Matrices for Run 5 Single Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net241	0.68	0.32	0.10	0.90	0.79
net242	0.74	0.26	0.08	0.92	0.83
net243	0.70	0.30	0.08	0.92	0.81
net244	0.66	0.34	0.08	0.92	0.79
net245	0.72	0.28	0.10	0.90	0.81
net246	0.72	0.28	0.08	0.92	0.82
net247	0.66	0.34	0.08	0.92	0.79
net248	0.76	0.24	0.10	0.90	0.83
net249	0.78	0.22	0.08	0.92	0.85
net250	0.68	0.32	0.08	0.92	0.80
net251	0.72	0.28	0.10	0.90	0.81
net252	0.68	0.32	0.06	0.94	0.81
net253	0.70	0.30	0.06	0.94	0.82
net254	0.70	0.30	0.06	0.94	0.82
net255	0.68	0.32	0.10	0.90	0.79
net256	0.76	0.24	0.08	0.92	0.84
net257	0.66	0.34	0.08	0.92	0.79
net258	0.72	0.28	0.12	0.88	0.80
net259	0.68	0.32	0.06	0.94	0.81
net260	0.64	0.36	0.08	0.92	0.78
net261	0.74	0.26	0.10	0.90	0.82
net262	0.70	0.30	0.10	0.90	0.80
net263	0.72	0.28	0.08	0.92	0.82
net264	0.66	0.34	0.10	0.90	0.78
net265	0.74	0.26	0.10	0.90	0.82
net266	0.70	0.30	0.08	0.92	0.81
net267	0.74	0.26	0.06	0.94	0.84
net268	0.72	0.28	0.10	0.90	0.81
net269	0.68	0.32	0.10	0.90	0.79
net270	0.72	0.28	0.06	0.94	0.83
Mean	0.7053	0.2947	0.0847	0.9153	0.8103
STD	0.0338	0.0338	0.0161	0.0161	0.0180

Table A.13. Observed Probability Matrices for Run 5 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
mvnet61	0.70	0.30	0.08	0.92	0.81
mvnet62	0.70	0.30	0.08	0.92	0.81
mvnet63	0.70	0.30	0.08	0.92	0.81
mvnet64	0.70	0.30	0.08	0.92	0.81
mvnet65	0.74	0.26	0.10	0.90	0.82
mvnet66	0.68	0.32	0.08	0.92	0.80
mvnet67	0.70	0.30	0.10	0.90	0.80
mvnet68	0.66	0.34	0.06	0.94	0.80
mvnet69	0.70	0.30	0.08	0.92	0.81
mvnet70	0.72	0.28	0.08	0.92	0.82
mvnet71	0.66	0.34	0.08	0.92	0.79
mvnet72	0.74	0.26	0.10	0.90	0.82
mvnet73	0.70	0.30	0.08	0.92	0.81
mvnet74	0.72	0.28	0.08	0.92	0.82
mvnet75	0.70	0.30	0.08	0.92	0.81
mvnet76	0.76	0.24	0.08	0.92	0.84
mvnet77	0.66	0.34	0.08	0.92	0.79
mvnet78	0.70	0.30	0.08	0.92	0.81
mvnet79	0.68	0.32	0.06	0.94	0.81
mvnet80	0.70	0.30	0.08	0.92	0.81
mvnet81	0.70	0.30	0.08	0.92	0.81
mvnet82	0.70	0.30	0.10	0.90	0.80
mvnet83	0.70	0.30	0.10	0.90	0.80
mvnet84	0.68	0.32	0.08	0.92	0.80
mvnet85	0.68	0.32	0.08	0.92	0.80
mvnet86	0.70	0.30	0.08	0.92	0.81
mvnet87	0.70	0.30	0.08	0.92	0.81
mvnet88	0.66	0.34	0.08	0.92	0.79
mvnet89	0.70	0.30	0.08	0.92	0.81
mvnet90	0.74	0.26	0.10	0.90	0.82
Mean	0.6993	0.3007	0.0827	0.9173	0.8083
STD	0.0239	0.0239	0.0100	0.0100	0.0104

Table A.14. Calculated Probability Matrices for Run 5 Majority Vote Nets

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P($ good $)$
cmvnet61	0.81	0.19	0.02	0.98	0.89
cmvnet62	0.78	0.22	0.02	0.98	0.88
cmvnet63	0.80	0.20	0.02	0.98	0.89
cmvnet64	0.82	0.18	0.02	0.98	0.90
cmvnet65	0.83	0.17	0.02	0.98	0.90
cmvnet66	0.77	0.23	0.02	0.98	0.87
cmvnet67	0.79	0.21	0.02	0.98	0.88
cmvnet68	0.78	0.22	0.01	0.99	0.88
cmvnet69	0.78	0.22	0.02	0.98	0.88
cmvnet70	0.81	0.19	0.02	0.98	0.90
cmvnet71	0.77	0.23	0.02	0.98	0.87
cmvnet72	0.82	0.18	0.02	0.98	0.90
cmvnet73	0.77	0.23	0.02	0.98	0.87
cmvnet74	0.78	0.22	0.02	0.98	0.88
cmvnet75	0.80	0.20	0.02	0.98	0.89
cmvnet76	0.83	0.17	0.02	0.98	0.91
cmvnet77	0.77	0.23	0.02	0.98	0.87
cmvnet78	0.80	0.20	0.02	0.98	0.89
cmvnet79	0.78	0.22	0.02	0.98	0.88
cmvnet80	0.79	0.21	0.02	0.98	0.89
cmvnet81	0.78	0.22	0.02	0.98	0.88
cmvnet82	0.80	0.20	0.02	0.98	0.89
cmvnet83	0.80	0.20	0.02	0.98	0.89
cmvnet84	0.76	0.24	0.02	0.98	0.87
cmvnet85	0.78	0.22	0.02	0.98	0.88
cmvnet86	0.81	0.19	0.02	0.98	0.90
cmvnet87	0.78	0.22	0.02	0.98	0.88
cmvnet88	0.78	0.22	0.01	0.99	0.89
cmvnet89	0.78	0.22	0.02	0.98	0.88
cmvnet90	0.82	0.18	0.02	0.98	0.90
Mean	0.7921	0.2079	0.0195	0.9805	0.8863
STD	0.0196	0.0196	0.0040	0.0040	0.0099

Table A.15. Incorrectly Classified Test Exemplars for Run 2 Through Run 5

Run 2S		Run 3S		Run 4S		Run 5S	
File	Count	File	Count	File	Count	File	Count
corr96	30	corr17	30	corr17	30	corr25	30
corr25	29	corr25	30	corr25	30	corr37	30
corr194	28	corr37	30	corr37	30	corr96	30
corr53	27	corr96	30	corr41	30	corr51	30
corr227	27	corr227	30	corr96	30	corr164	30
corr164	26	corr194	30	corr194	30	corr182	30
corr7	24	corr53	29	corr168	29	corr194	30
corr182	24	corr7	28	corr53	28	corr29	29
corr37	23	corr51	28	corr164	28	corr53	29
corr145	23	corr164	28	corr182	28	corr170	29
corr170	22	corr182	28	corr227	28	corr241	29
corr168	20	corr168	26	corr7	27	corr17	27
corr17	17	corr169	26	corr19	26	corr19	27
corr92	16	corr145	25	corr170	25	corr145	27
corr241	16	corr41	24	corr51	23	corr227	26
corr82	15	corr241	21	corr152	22	corr7	22
corr19	13	corr29	20	corr176	15	corr57	21
corr31	12	corr19	19	corr241	15	corr41	19
corr57	12	corr57	17	corr29	14	corr168	19
corr90	11	corr82	12	corr82	10	corr176	19
corr41	10	corr35	9	corr200	10	corr82	13
corr80	10	corr39	7	corr57	9	corr200	7
corr86	10	corr176	7	corr31	8	corr31	4
corr88	10	corr205	4	corr35	5	corr39	4
corr64	9	corr200	4	corr80	3	corr151	3
corr66	9	corr31	3	corr39	3	corr35	2
corr51	9	corr128	3	corr43	3	corr45	1
corr35	8	corr27	2	corr190	3	corr219	1
corr84	8	corr190	2	corr27	2	corr190	1
corr62	7	corr13	1	corr184	1	corr202	1
corr70	7	corr43	1				
corr254	7	corr59	1				
corr29	7	corr178	1				
corr27	6	corr201	1				
corr176	5						
corr243	5						
corr68	4						
corr72	4						
corr190	4						
corr205	4						
corr233	4						
corr149	3						
corr200	3						
corr202	3						
corr231	3						
corr235	3						
corr43	2						
corr147	2						
corr39	1						
corr151	1						
corr211	1						
corr213	1						
corr223	1						
corr225	1						

Appendix B. *Data File Samples and Processing Software*

B.1 Preprocessing of Correlation Product Data Files.

The actual preprocessing of the correlation product data was done using a commercial digital signal processing package called DADiSP Worksheet^{tm}, by DSP Developement Corporation, One Kendall Square, Cambridge, MA 02139. The software package is a graphics-based spreadsheet with a multi-window environment. The package has its own Command File language for automating processing tasks. For completeness, the command line of the windows of the worksheet are shown here, followed by a sample of a command file used to process the correlation data. Figures B.1 and B.2 show the before and after plots of a typical direct sequence correlation signature, while figures B.3 and B.4 show before and after plots of a typical frequency hopped signature.

```
DADiSP Worksheet$\,^{tm}$ algorithm implemented in a
worksheet called REDUCE1.

WINDOW 1 : <file read in here>
WINDOW 2 : Decimate(W1,2,1)
WINDOW 3 : Decimate(W1,2,2)
WINDOW 4 : Avg(W2,W3)
WINDOW 5 : Abs(W4) | fmax
WINDOW 6 : W4/getpt(W5,curpos(W5))  |  fmax  | nmove(-25)
WINDOW 7 : Extract(W6,curpos(W6),50)
```

Sample of DADiSP Worksheet^{tm} Command File.

```
D D:\corrdat @cr O thesis @cr W L reduce1 @cr E
@cntl_home
@f8 CORR6.1 @cr @cr writea("fcorr6.dat",w7) @cr
@f8 CORR7.1 @cr @cr writea("fcorr7.dat",w7) @cr
@f8 CORR9.1 @cr @cr writea("fcorr9.dat",w7) @cr
@f8 CORR10.1 @cr @cr writea("fcorr10.dat",w7) @cr
@f8 CORR12.1 @cr @cr writea("fcorr12.dat",w7) @cr
@f8 CORR13.1 @cr @cr writea("fcorr13.dat",w7) @cr
@f8 CORR14.1 @cr @cr writea("fcorr14.dat",w7) @cr
@f8 CORR15.1 @cr @cr writea("fcorr15.dat",w7) @cr
@f8 CORR16.1 @cr @cr writea("fcorr16.dat",w7) @cr
@f8 CORR17.1 @cr @cr writea("fcorr17.dat",w7) @cr

  (Same pattern repeated for each corrXX file)

@f8 CORR52.1 @cr @cr writea("fcorr52.dat",w7) @cr
@f8 CORR53.1 @cr @cr writea("fcorr53.dat",w7) @cr
@f8 CORR54.1 @cr @cr writea("fcorr54.dat",w7) @cr
@f8 CORR55.8 @cr @cr writea("fcorr55.dat",w7) @cr
@f8 CORR56.1 @cr @cr writea("fcorr56.dat",w7) @cr
@f8 CORR57.1 @cr @cr writea("fcorr57.dat",w7) @cr
@f8 CORR58.1 @cr @cr writea("fcorr58.dat",w7) @cr
@f8 CORR59.1 @cr @cr writea("fcorr59.dat",w7) @cr
@f8 CORR60.1 @cr @cr writea("fcorr60.dat",w7) @cr
@esc @esc @esc y @esc @esc @esc
```

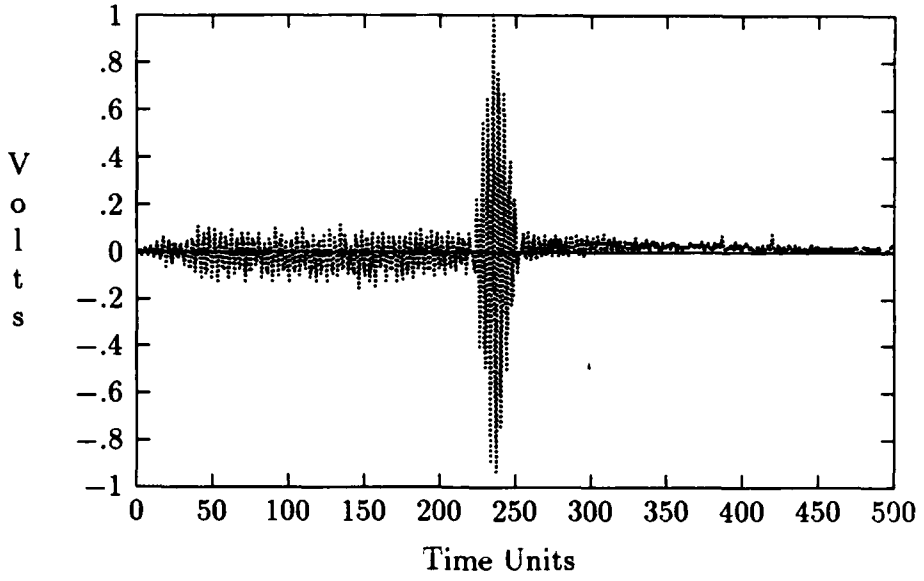

Figure B.1. Direct Sequence Correlation Product CORR18 Before Processing

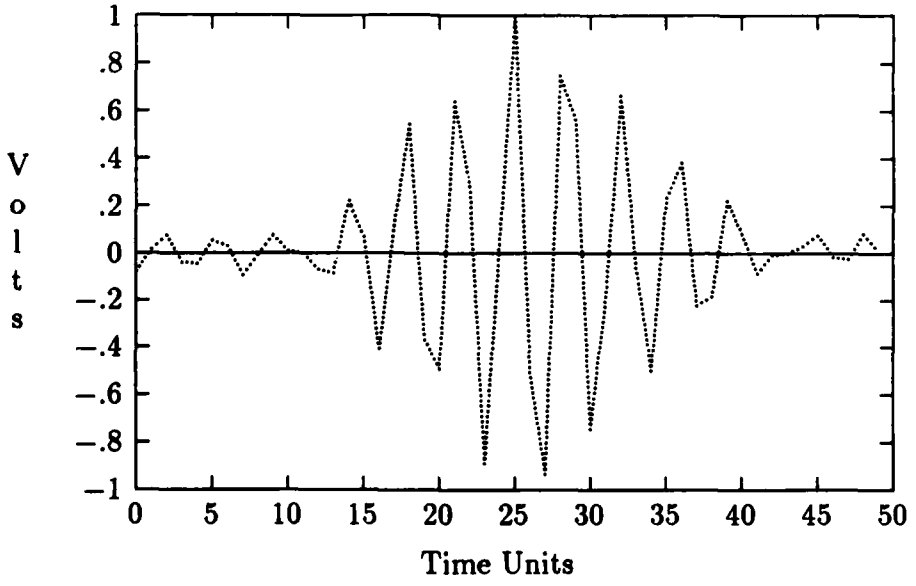

Figure B.2. Direct Sequence Correlation Product CORR18 After Processing

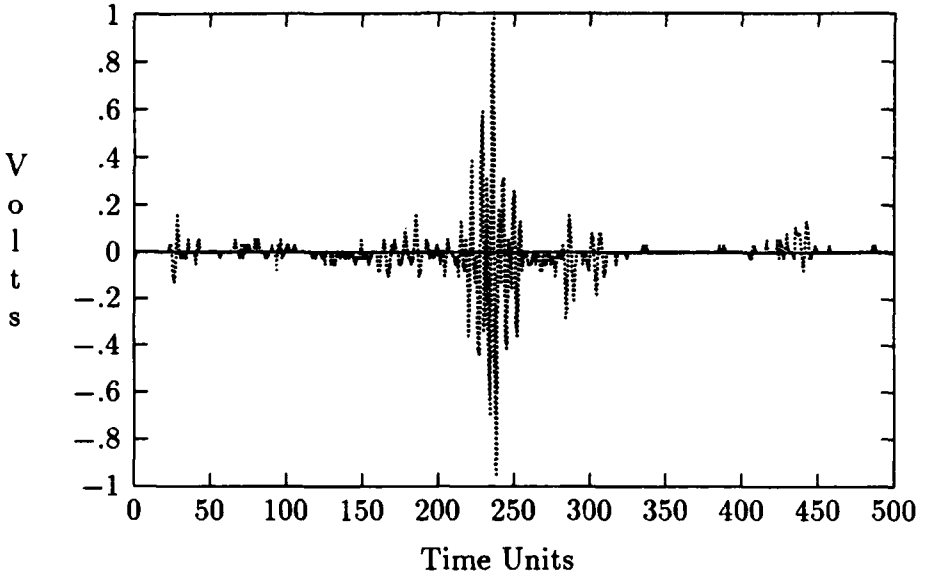

Figure B.3. Frequency-Hopped Correlation Product CORR148 After Processing

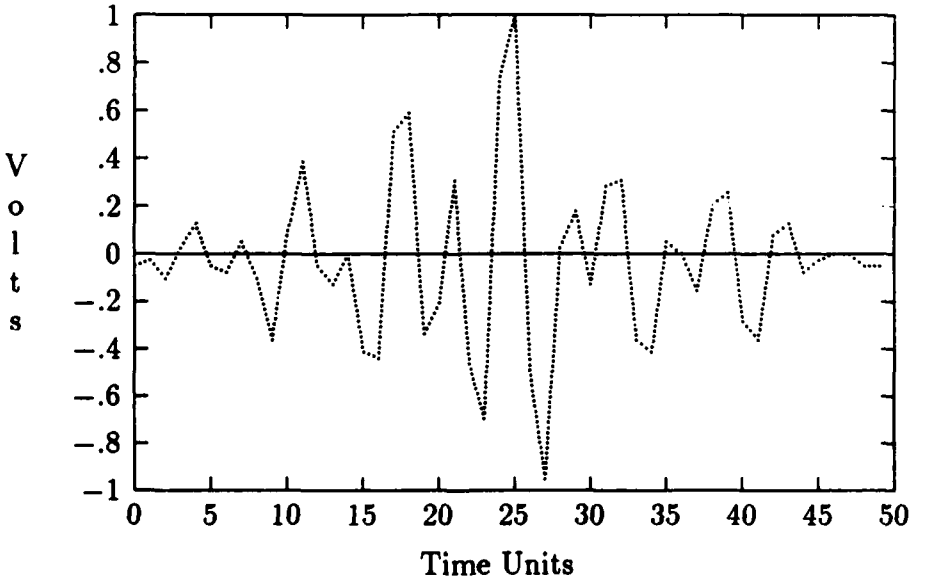

Figure B.4. Frequency-Hopped Correlation Product CORR148 After Processing

B.2 Construction of Datasets.

In this section, the details of how the datasets used to train the networks, were constructed. The software routine is written in QuickBasic^{tm}. The user is prompted for the number of exemplars to be used for training, the number of exemplars for testing, the number of classes of exemplars, and the number of elements in each exemplar. The routine expects the user to provide the name of an input file containing the a sequence number, filename, and class for each correlation signature to be included in the dataset. This provides for absolute control over the exact structure and mix of exemplars in the training and test datasets. The NeuralGraphics simulator reserves the specified number of exemplars for testing from the bottom of the file. In other words, if 100 test exemplars are specified for a file containing a total of 250 exemplars, the last 100 exemplars will be used as the test dataset. Presented here is a sample input file and the source code for the routine CONSTRUC.BAS.

Sample input file for constucting a dataset.

```
1, "d:\data\corrdat1\fcorr6.dat", 1
2, "d:\data\corrdat4\fcorr61.dat", 2
3, "d:\data\corrdat1\fcorr9.dat", 1
4, "d:\data\corrdat4\fcorr63.dat", 2
5, "d:\data\corrdat1\fcorr12.dat", 1
6, "d:\data\corrdat4\fcorr65.dat", 2
7, "d:\data\corrdat1\fcorr14.dat", 1
8, "d:\data\corrdat4\fcorr67.dat", 2
9, "d:\data\corrdat1\fcorr16.dat", 1
10, "d:\data\corrdat4\fcorr69.dat", 2
```

(Same pattern repeated for each file included in dataset)

```
192, "d:\data\corrdat4\fcorr233.dat", 2
193, "d:\data\corrdat1\fcorr194.dat", 1
194, "d:\data\corrdat4\fcorr235.dat", 2
195, "d:\data\corrdat1\fcorr196.dat", 1
196, "d:\data\corrdat4\fcorr237.dat", 2
197, "d:\data\corrdat1\fcorr198.dat", 1
198, "d:\data\corrdat4\fcorr239.dat", 2
199, "d:\data\corrdat1\fcorr200.dat", 1
200, "d:\data\corrdat4\fcorr241.dat", 2
201, "d:\data\corrdat1\fcorr202.dat", 1
202, "d:\data\corrdat4\fcorr243.dat", 2
```

Source Code for CONSTRUC

```
'Program: CONSTRUC.BAS
'Author: John W. DeBerry
'Description:  This routine reads a ASCII data file called
'names$ consisting of multiple lines of a sequence number,
'a filename, and a class number. The filenames contain 50
'element correlation product vectors in a column. Each vector
'is read and then written to a super data file of the name
'specified by the user. The format of the super file is:

'     Sequence #1, element(0), element(1), ..., element(49)
'     Class #
'     Sequence #2, element(0), element(1), ..., element(99)
'     Class #
'     Sequence #3, element(0), element(1), ..., element(99)
'     Class #
'           .
'           .
'           .
'     EOF
'
'This super file will be used as an input file for the
'NeuralGraphics simulator written by Greg Tarr.

INPUT "What is the file containing the name & class data"; names$
CLS
```

```
INPUT "What shall I name the super data file"; super$
CLS
INPUT "How many files are to be training vectors"; exemplars%
CLS
INPUT "How many are to be test vectors"; texemplars%
CLS
INPUT "How many classes of vectors are there"; outelements%
CLS
INPUT "How many elements per vector"; inelements%
CLS
DIM vector!(inelements%)
PRINT "Name file - "; names$
PRINT "Super file - "; super$
PRINT
PRINT "File being processed - ";
OPEN super$ FOR OUTPUT AS #1
OPEN names$ FOR INPUT AS #2
PRINT #1, exemplars%; texemplars%; inelements%; outelements%

DO UNTIL EOF(2)
   INPUT #2, number%, file$, class%
   LOCATE 4, 26
   PRINT file$
   OPEN file$ FOR INPUT AS #3
   FOR i = 0 TO (inelements% - 1)
      INPUT #3, vector!(i)
   NEXT i
   CLOSE #3
   PRINT #1, number%;
   FOR i = 0 TO (inelements% - 1)
      PRINT #1, vector!(i); " ";
   NEXT i
   PRINT #1,
   PRINT #1, class%
LOOP
CLOSE
END
```

B.3 Processing of NeuralGraphics Output.

In this section, samples of the actual data file output of the NeuralGraphics software will be shown; one file containing the results of testing a network with the test data, and one file containing the training history data. Following that the QuickBasic source code for several routines used to process the data are shown. The routine PTABLE_2.BAS and MV3TABLE.BAS operate on the network performance data, while GOOD.BAS operates on the history files. The routine THEORY.BAS operates on the output files of PTABLE_2.BAS. The first three routines expect two things, a file containing a list of filenames to operate on, and the actual files specified by that list. THEORY.BAS uses the names in the list as names for the $\underline{P}$ matrices it constructs from the matrices found in the matrix table it operates on. This was done just to keep from mixing up the matrices.

```
Sample output file from NeuralGraphics simulator

net10.dat

Exemplar #  True Class  Net Decision
103          1          2
104          2          2
105          1          1
106          2          2
107          1          1
108          2          2
109          1          1
110          2          2
111          1          2
112          2          2
113          1          2
```

114	2	2
115	1	1
116	2	2
117	1	1
118	2	2
119	1	2
120	2	2
121	1	1
122	2	2
123	1	1
124	2	2
125	1	1
126	2	2
127	1	1
128	2	2
129	1	1
130	2	2
131	1	2
132	2	2
133	1	1
134	2	2
135	1	2
136	2	2
137	1	1
138	2	2
139	1	1
140	2	2
141	1	1
142	2	2
143	1	1
144	2	1
145	1	2
146	2	2
147	1	2
148	2	2
149	1	1
150	2	2
151	1	2
152	2	1
153	1	1
154	2	2
155	1	1

156	2	2
157	1	1
158	2	2
159	1	1
160	2	2
161	1	1
162	2	2
163	1	2
164	2	2
165	1	1
166	2	2
167	1	2
168	2	2
169	1	2
170	2	2
171	1	1
172	2	2
173	1	1
174	2	2
175	1	1
176	2	2
177	1	1
178	2	2
179	1	1
180	2	2
181	1	2
182	2	2
183	1	1
184	2	2
185	1	1
186	2	1
187	1	1
188	2	2
189	1	1
190	2	2
191	1	1
192	2	2
193	1	2
194	2	2
195	1	1
196	2	2
197	1	1

198	2	2
199	1	1
200	2	1
201	1	1
202	2	2
EOF		

Sample history file produced by NeuralGraphics simulator

Training history - net10

Seed - 620962919

Count	Error	Training		Test		#learned
		Right	Good	Right	Good	
1000	7.61	11.80	48.80	30.00	62.00	0
2000	5.53	68.50	85.90	65.00	79.00	1
3000	3.64	84.50	92.40	73.00	82.00	25
4000	1.83	96.20	98.70	76.00	84.00	78
5000	1.23	97.20	99.20	76.00	82.00	93
6000	0.43	99.70	100.00	78.00	81.00	99
7000	0.24	100.00	100.00	79.00	81.00	100
8000	0.17	100.00	100.00	79.00	82.00	100
9000	0.15	100.00	100.00	79.00	82.00	100
10000	0.13	100.00	100.00	79.00	82.00	100
11000	0.12	100.00	100.00	80.00	82.00	100
12000	0.11	100.00	100.00	80.00	82.00	100
13000	0.11	100.00	100.00	80.00	82.00	100
14000	0.10	100.00	100.00	79.00	82.00	100
15000	0.10	100.00	100.00	80.00	82.00	100
16000	0.09	100.00	100.00	80.00	82.00	100
17000	0.09	100.00	100.00	80.00	82.00	100
18000	0.09	100.00	100.00	80.00	82.00	100
19000	0.08	100.00	100.00	80.00	82.00	100
20000	0.08	100.00	100.00	80.00	82.00	100

EOF

```
'Program: PTABLE_2.BAS
'Author: John W. DeBerry
'This routine reads the data file containing the final
'classifications yielded by a net on the test set exemplars.
'It then calculates the P matrix for that net. Multiplying
'the values of the P matrix by 100 yields the actual observed
'percent correct (or wrong) performance of the net.  The
'routine writes this info to a file in rows for each net
'data file in the name$ file. If the user specifies LOTUS
'format, the result is a table ready for import into LOTUS
'for computing the average and STD if each column ( P(1/1),
'P(2/1), P(1/2), P(2/2), and P(good) )

OPTION BASE 1
REM $DYNAMIC
LOCATE 23, 2
INPUT "What source file for the data file names"; name$
CLS
LOCATE 23, 2
INPUT "Filename for probability matrix table"; matrix$
CLS
LOCATE 23, 2
INPUT "Filename for out of class vector log"; verror$
CLS
DO
   LOCATE 23, 2
   INPUT "Do you want Lotus type file"; a$
LOOP UNTIL a$ = "y" OR a$ = "n"
CLS
LOCATE 12, 31
PRINT "WORKING ....."
DIM testinfo%(3), cerror%(2), vectornum%(50), classcount%(2)
DIM prob!(5)
OPEN matrix$ FOR OUTPUT AS #1
OPEN name$ FOR INPUT AS #2
OPEN verror$ FOR OUTPUT AS #3
IF a$ = "n" THEN
PRINT #1 "Net ID        P(1|1)   P(2|1)   P(1|2)   P(2|2)  P(good)"
PRINT #1,"-------------------------------------------------------"
END IF
PRINT #3 "Net ID        Out - of - Class vectors"
PRINT #3,"-------------------------------------------------------"
```

```
DO UNTIL EOF(2)
   FOR i = 1 TO 2
      cerror%(i) = 0
      classcount%(i) = 0
   NEXT i
   wrong% = 0
   INPUT #2, net$
   OPEN net$ FOR INPUT AS #4
   LINE INPUT #4, junk$
   DO UNTIL EOF(4)
      INPUT #4, testinfo%(1), testinfo%(2), testinfo%(3)
      IF testinfo%(2) = 1 THEN
         classcount%(1) = classcount%(1) + 1
         IF testinfo%(2) <> testinfo%(3) THEN
            wrong% = wrong% + 1
            vectornum%(wrong%) = testinfo%(1)
            cerror%(1) = cerror%(1) + 1
         END IF
      ELSEIF testinfo%(2) = 2 THEN
         classcount%(2) = classcount%(2) + 1
         IF testinfo%(2) <> testinfo%(3) THEN
            wrong% = wrong% + 1
            vectornum%(wrong%) = testinfo%(1)
            cerror%(2) = cerror%(2) + 1
         END IF
      END IF
   LOOP
   CLOSE #4
   count% = classcount%(1) + classcount%(2)
   prob!(2) = cerror%(1) / classcount%(1)
   prob!(3) = cerror%(2) / classcount%(2)
   prob!(1) = 1 - prob!(2)
   prob!(4) = 1 - prob!(3)
   prob!(5) = (classcount%(1)/count%)*prob!(1)+
                     (classcount%(2)/count%)*prob!(4)
   PRINT #3, LEFT$(net$, LEN(net$) - 4);
   IF LEN(net$) - 4 = 4 THEN PRINT #3, SPC(4);
   IF LEN(net$) - 4 = 5 THEN PRINT #3, SPC(3);
   IF LEN(net$) - 4 = 6 THEN PRINT #3, SPC(2);
   FOR i = 1 TO wrong%
      PRINT #3, vectornum%(i);
   NEXT i
```

```
      PRINT #3,
      IF a$ = "n" THEN
         PRINT #1, LEFT$(net$, LEN(net$) - 4),
         FOR i = 1 TO 5
            PRINT #1, prob!(i); SPC(4);
         NEXT i
         PRINT #1,
      ELSEIF a$ = "y" THEN
         PRINT #1, CHR$(34); LEFT$(net$, LEN(net$) - 4); CHR$(34);
         FOR i = 1 TO 5
            PRINT #1, ","; prob!(i);
         NEXT i
         PRINT #1,
      END IF
   LOOP
   CLS
   CLOSE
   END
```

```
'Program: MV3TABLE.BAS
'Author: John W. DeBerry
'This routine takes three net decision data files and
'contructs the P matrix for the equivalent majority
'vote network.  The P matrix for this equivalent network
'is then written in row format to matrix$. The matrix$
'file can be imported directly into LOTUS.


OPTION BASE 1
REM $DYNAMIC
LOCATE 23, 2
INPUT "What source file for majority vote file names"; name$
CLS
LOCATE 23, 2
INPUT "Filename for probability matrix table"; matrix$
CLS
LOCATE 23, 2
INPUT "Filename for out of class vector log"; verror$
CLS
DO
   LOCATE 23, 2
   INPUT "Do you want Lotus type file"; a$
LOOP UNTIL a$ = "y" OR a$ = "n"
CLS
LOCATE 12, 31
PRINT "WORKING ....."
DIM testinfo%(3), cerror%(2), vectornum%(50), classcount%(2)
DIM testinfo1%(3), testinfo2%(3), testinfo3%(3)
DIM prob!(5)
OPEN matrix$ FOR OUTPUT AS #1
OPEN name$ FOR INPUT AS #2
OPEN verror$ FOR OUTPUT AS #3
IF a$ = "n" THEN
 PRINT #1,"Net ID        P(1|1)   P(2|1)   P(1|2)   P(2|2)  P(good)"
 PRINT #1,"-------------------------------------------------------"
END IF
PRINT #3,"Net ID        Out - of - Class vectors"
PRINT #3,"-----------------------------------------------------"
DO UNTIL EOF(2)
   FOR i = 1 TO 2
      cerror%(i) = 0
```

```
   classcount%(i) = 0
NEXT i
wrong% = 0
INPUT #2, net1$, net2$, net3$, mvnet$
OPEN net1$ FOR INPUT AS #4
OPEN net2$ FOR INPUT AS #5
OPEN net3$ FOR INPUT AS #6
LINE INPUT #4, junk$
LINE INPUT #5, junk$
LINE INPUT #6, junk$
DO UNTIL EOF(6)
   INPUT #4, testinfo1%(1), testinfo1%(2), testinfo1%(3)
   INPUT #5, testinfo2%(1), testinfo2%(2), testinfo2%(3)
   INPUT #6, testinfo3%(1), testinfo3%(2), testinfo3%(3)
   testinfo%(1) = testinfo1%(1)
   testinfo%(2) = testinfo1%(2)
   vote% = testinfo1%(3) + testinfo2%(3) + testinfo3%(3)
   IF vote% <= 4 THEN
      testinfo%(3) = 1
   ELSEIF vote% >= 5 THEN
      testinfo%(3) = 2
   END IF
   IF testinfo%(2) = 1 THEN
      classcount%(1) = classcount%(1) + 1
      IF testinfo%(2) <> testinfo%(3) THEN
         wrong% = wrong% + 1
         vectornum%(wrong%) = testinfo%(1)
         cerror%(1) = cerror%(1) + 1
      END IF
   ELSEIF testinfo%(2) = 2 THEN
      classcount%(2) = classcount%(2) + 1
      IF testinfo%(2) <> testinfo%(3) THEN
         wrong% = wrong% + 1
         vectornum%(wrong%) = testinfo%(1)
         cerror%(2) = cerror%(2) + 1
      END IF
   END IF
LOOP
CLOSE #4
CLOSE #5
CLOSE #6
count% = classcount%(1) + classcount%(2)
```

```
   prob!(2) = cerror%(1) / classcount%(1)
   prob!(3) = cerror%(2) / classcount%(2)
   prob!(1) = 1 - prob!(2)
   prob!(4) = 1 - prob!(3)
   prob!(5) = (classcount%(1)/count%)*prob!(1)+
                              (classcount%(2)/count)*prob!(4)
   PRINT #3, LEFT$(mvnet$, LEN(mvnet$) - 4);
   IF LEN(mvnet$) - 4 = 6 THEN PRINT #3, SPC(4);
   IF LEN(mvnet$) - 4 = 7 THEN PRINT #3, SPC(3);
   IF LEN(mvnet$) - 4 = 8 THEN PRINT #3, SPC(2);
   FOR i = 1 TO wrong%
      PRINT #3, vectornum%(i);
   NEXT i
   PRINT #3,
   IF a$ = "n" THEN
      PRINT #1, LEFT$(mvnet$, LEN(mvnet$) - 4),
      FOR i = 1 TO 5
         PRINT #1, prob!(i); SPC(4);
      NEXT i
      PRINT #1,
   ELSEIF a$ = "y" THEN
      PRINT #1, CHR$(34); LEFT$(mvnet$, LEN(mvnet$) - 4); CHR$(34);
      FOR i = 1 TO 5
         PRINT #1, ","; prob!(i);
      NEXT i
      PRINT #1,
   END IF
LOOP
CLS
CLOSE
END
```

```
'Program: MVTHEORY.BAS
'Author: John W. DeBerry
'This routine constructs the THEORETICAL majority vote
'P matrix from the P matrices of three nets. The computations
'assume independence. The output file matrix$ is row oriented
'and can be directly imported into LOTUS. The three nets used
'to generate the majority vote matrix are read from name$. The
'first majority vote net is constructed from the first three
'net matrices in the single net P matrix table, the next from
'the next three matrices in the table, etc. The total number of
'rows in the single net matrix table should be divisible by three.
'The program reads netname$ for assigning a name to the majority
'vote nets (for flexibility of filenaming). The user must know
'the a priori probabilities of the test set exemplars.

OPTION BASE 1
REM $DYNAMIC
LOCATE 23, 2
INPUT "What source file for single net matrix"; name$
CLS
LOCATE 23, 2
INPUT "Filename for probability matrix table"; matrix$
CLS
LOCATE 23, 2
INPUT "What are the a priori probabilities (class 1, class2)"; p1, p2
LOCATE 23, 2
INPUT "What is the file containing the names of the mvnets"; netname$

LOCATE 12, 31
PRINT "WORKING ....."
DIM n1(5), n2(5), n3(5)
DIM prob!(5)
OPEN matrix$ FOR OUTPUT AS #1
OPEN name$ FOR INPUT AS #2
OPEN netname$ FOR INPUT AS #3
DO UNTIL EOF(2)
 INPUT #3, mvnet$
 INPUT #2, junk$, n1(1), n1(2), n1(3), n1(4), n1(5)
 INPUT #2, junk$, n2(1), n2(2), n2(3), n2(4), n2(5)
 INPUT #2, junk$, n3(1), n3(2), n3(3), n3(4), n3(5)
 prob!(1)=n1(1)*n2(1)*n3(1)+n1(1)*n2(1)*n3(2)+n1(1)*n2(2)*n3(1)+
                  n1(2)*n2(1)*n3(1)
```

```
 prob!(2)=1 - prob!(1)
 prob!(4)=n1(4)*n2(4)*n3(4)+n1(4)*n2(4)*n3(3)+n1(4)*n2(3)*n3(4)+
                  n1(3)*n2(4)*n3(4)

 prob!(3)=1 - prob!(4)

 prob!(5)=p1*prob!(1) + p2*prob!(4)
      PRINT #1, CHR$(34); mvnet$; CHR$(34);
      FOR i = 1 TO 5
         PRINT #1, ","; prob!(i);
      NEXT i
      PRINT #1,
LOOP
CLS
CLOSE
END
```

```
'Program: MVTHEORY.BAS
'Author: John W. DeBerry
'This routine reads stripped output from the NeuralGraphics
'simulator and reads only the desired information. It writes
'the P(good) history of each net in rows to a file training$.
'The rows of each P(good) history for each net listed in the
'file filename$ will be written. The training$ file can then
'be used with LOTUS to get the average and STD value at each
'1000 iteration interval up to 20000.
'
'
DIM count(20), aerr!(20), tright!(20), tgood!(20), right!(20),
    good!(20), learn(20)
INPUT "What filename contains the history file names"; filename$
CLS
INPUT "What filename for the processed data"; training$
CLS
OPEN filename$ FOR INPUT AS #1
OPEN training$ FOR OUTPUT AS #3

PRINT #3, CHR$(34); "Count"; CHR$(34);
PRINT #3,
DO UNTIL EOF(1)
   INPUT #1, name$
   OPEN name$ FOR INPUT AS #2

   FOR i = 1 TO 5
      LINE INPUT #2, stuff$
   NEXT i
   FOR i = 0 TO 19
      INPUT #2, count(i),aerr!(i),tright!(i),tgood!(i),right!(i),
                good!(i),learn(i)
   NEXT i
   PRINT #3, CHR$(34); LEFT$(name$, LEN(name$) - 4); CHR$(34);
   FOR i = 0 TO 19
      PRINT #3, good!(i);
   NEXT i
   PRINT #3,
   CLOSE #2
LOOP
CLOSE
END
```

Appendix C. *Table of Percentage Points of the Wilk-Shapiro Statistic (reproduced from [14])*

Table 6. *Percentage points of the W test* for n = 3(1)50*

	Level								
n	0·01	0·02	0·05	0·10	0·50	0·90	0·95	0·98	0·99
3	0·753	0·756	0·767	0·789	0·959	0·998	0·999	1·000	1·000
4	·687	·707	·748	·792	·935	·987	·992	·996	·997
5	·686	·715	·762	·806	·927	·979	·986	·991	·993
6	0·713	0·743	0·788	0·826	0·927	0·974	0·981	0·986	0·989
7	·730	·760	·803	·838	·928	·972	·979	·985	·988
8	·749	·778	·818	·851	·932	·972	·978	·984	·987
9	·764	·791	·829	·859	·935	·972	·978	·984	·986
10	·781	·806	·842	·869	·938	·972	·978	·983	·986
11	0·792	0·817	0·850	0·876	0·940	0·973	0·979	0·984	0·986
12	·805	·828	·859	·883	·943	·973	·979	·984	·986
13	·814	·837	·866	·889	·945	·974	·979	·984	·986
14	·825	·846	·874	·895	·947	·975	·980	·984	·986
15	·835	·855	·881	·901	·950	·975	·980	·984	·987
16	0·844	0·863	0·887	0·906	0·952	0·976	0·981	0·985	0·987
17	·851	·869	·892	·910	·954	·977	·981	·985	·987
18	·858	·874	·897	·914	·956	·978	·982	·986	·988
19	·863	·879	·901	·917	·957	·978	·982	·986	·988
20	·868	·884	·905	·920	·959	·979	·983	·986	·988
21	0·873	0·888	0·908	0·923	0·960	0·980	0·983	0·987	0·989
22	·878	·892	·911	·926	·961	·980	·984	·987	·989
23	·881	·895	·914	·928	·962	·981	·984	·987	·989
24	·884	·898	·916	·930	·963	·981	·984	·987	·989
25	·888	·901	·918	·931	·964	·981	·985	·988	·989
26	0·891	0·904	0·920	0·933	0·965	0·982	0·985	0·988	0·989
27	·894	·906	·923	·935	·965	·982	·985	·988	·990
28	·896	·908	·924	·936	·966	·982	·985	·988	·990
29	·898	·910	·926	·937	·966	·982	·985	·988	·990
30	·900	·912	·927	·939	·967	·983	·985	·988	·900
31	0·902	0·914	0·929	0·940	0·967	0·983	0·986	0·988	0·990
32	·904	·915	·930	·941	·968	·983	·986	·988	·990
33	·906	·917	·931	·942	·968	·983	·986	·989	·990
34	·908	·919	·933	·943	·969	·983	·986	·989	·990
35	·910	·920	·934	·944	·969	·984	·986	·989	·990
36	0·912	0·922	0·935	0·945	0·970	0·984	0·986	0·989	0·990
37	·914	·924	·936	·946	·970	·984	·987	·989	·990
38	·916	·925	·938	·947	·971	·984	·987	·989	·990
39	·917	·927	·939	·948	·971	·984	·987	·989	·991
40	·919	·928	·940	·949	·972	·985	·987	·989	·991
41	0·920	0·929	0·941	0·950	0·972	0·985	0·987	0·989	0·991
42	·922	·930	·942	·951	·972	·985	·987	·989	·991
43	·923	·932	·943	·951	·973	·985	·987	·990	·991
44	·924	·933	·944	·952	·973	·985	·987	·990	·991
45	·926	·934	·945	·953	·973	·985	·988	·990	·991
46	0·927	0·935	0·945	0·953	0·974	0·985	0·988	0·990	0·991
47	·928	·936	·946	·954	·974	·985	·988	·990	·991
48	·929	·937	·947	·954	·974	·985	·988	·990	·991
49	·929	·937	·947	·955	·974	·985	·988	·990	·991
50	·930	·938	·947	·955	·974	·985	·988	·990	·991

* Based on fitted Johnson (1949) S_B approximation, see Shapiro & Wilk (1965a) for details.

Bibliography

1. R. L. Pickholtz *et al.*, "Theory of Spread-Spectrum Communication – a Tutorial," *IEEE Transactions on Communications*, vol. 30, pp. 855–884, May 1982.
2. C. Garvin, Personal Correspondence. U.S. Army, Harry Diamond Laboratories, Adelphi, Maryland, January – September 1989.
3. R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4–22, April 1987.
4. S. K. Rogers and M. Kabrisky, "Biological and Artificial Neural Networks for Pattern Recognition." School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, 1989. Short Course Notes.
5. "Darpa Neural Network Study, October 1987 - February 1988, Executive Summary." Lincoln Laboratory, Massachusetts Institute of Technology, Lexington Massachusetts, July 1988.
6. S. E. Troxel *et al.*, "The Use of Neural Networks in PSRI Target Recognition," in *IEEE International Conference on Neural Networks*, pp. 593–600, 1988.
7. R. P. Gorman and T. J. Sejnowski, "Learned Classification of Sonar Targets Using a Massively Parallel Network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1135–1140, July 1988.
8. D. W. Ruck, "Multisensor Target Detection and Classification," Master's thesis, AFIT/GE/ENG/87D-56. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1987.
9. C. C. Piazza, "Modified Backward Error Propagation for Tactical Target Recognition," Master's thesis, AFIT/GE/ENG/88D-36. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.
10. M. K. Lutey, "Problem Specific Applications for Neural Networks," Master's thesis, AFIT/GE/ENG/88D-23. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.
11. G. L. Tarr, "Dynamic Analysis of Feedforward Neural Networks Using Simulated and Measured Data," Master's thesis, AFIT/GE/ENG/88D-54. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.
12. R. W. Hamming, *Coding and Information Theory.* Prentice-Hall, second ed., 1986.
13. NH Analytical Software, 1958 Eldrige Avenue, Roseville MN 55113, *Statistix™, An Interactive Statistical Analysis Program for Microcomputers.*

14. S. S. Shapiro and M. B. Wilk, "An Analysis of Variance Test for Normality," *Biometrika*, vol. 52, pp. 591–611, 1965.

15. S. D. Schlotzhauer and R. C. Littell, *SAS^{tm} System for Elementary Statistical Analysis.* SAS Institute Inc., Box 8000, SAS Circle, Cary NC 27512-8000.

16. I. Miller and J. E. Freund, *Probability and Statistics for Engineers.* Prentice-Hall, third ed., 1985.

Vita

Captain John W. DeBerry [redacted]

[redacted] He attended Erskine College in Due West, South Carolina and received a Bachelor of Science degree in biology in 1978. He enlisted into the United States Air Force as an officer trainee in 1982. After completion of Officer Training School, he received his commission on December 22, 1982. He was accepted into the Air Force Institute of Technology in residence undergraduate degree conversion program and received a Bachelor of Science degree in Electrical Engineering in March 1985. He was assigned to the 1815th Operational Test and Evaluation Squadron where he served as the Team Chief of a high frequency technical evaluation team until April 1988. In May of 1988, he again entered the School of Engineering, Air Force Institute of Technology.

[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/89D-10	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION US Army Harry Diamond Labs 8b. OFFICE SYMBOL (If applicable) SLC-HD-ST-OP	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 2800 Powder Mill Rd Adelphi MD 20783	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Classification of Acousto-Optic Correlation Signatures of Spread-Spectrum Signals Using Artificial Neural Networks

12. PERSONAL AUTHOR(S)
John W. DeBerry, B.S., B.S.E.E., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Thesis	FROM _____ TO _____	1989 December	121

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Neural Networks, Spread-Spectrum, Signal Classification, Pattern Recognition, Backward Error Propagation
12	09		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The primary goal of this reasearch was to determine if Artificial Neural Networks (ANNs) can be trained to classify the correlation signatures of direct sequence and frequency-hopped spread-spectrum signals. Secondary goals are to (1) determine if network classification performance can be modeled with a conditional probability matrix (2) determine if the symmetry of the matrices can be controlled, and (3) determine if using a majority vote rule over independently trained networks improves classification performance.

Correlation signatures of the two types of spread-spectrum signals were obtained from United States Army Harry Diamond Laboratories. These signatures were preprocessed and separated into various training and testing data sets. Thirty samples of network responses for several sets of training conditions were gathered using a neural network simulator.

ANNs trained directly on correlation signature data yielded classification accuracies on test data at or near 80%. The probability matrices were

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
David M. Norman, LtCol, USAF	(513) 255-9267	AFIT/ENG

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

BLOCK 19 CONTINUED

stationary with regard to test sets and the ability to shift the symmetry of the matrices was demonstrated. Improvement of classification accuracy via majority vote was possible if the nets were trained on different data sets. An average improvement of 1.8% was found to be statistically significant at a confidence level of 0.05. A metric was developed to estimate the similarity of the solutions found by networks in a given training run.